January 1989 Vol. 2 Nº 4 Price £1.80

Archive

The Subscription Magazine for Archimedes Users

RISC-OS Special – Power Released!!!

SpellCheck for Pipedream

Using Logistix

Current Directory Module / BASIC Compactor

Disc Recovery Update / Debugger's Breakpoints

WIMP Chess Review / Deputy Comms Review

Cobra Snake Game & Pax Art Package Reviews

Plus… *Competition Time!!!*

RISC-OS Special

Here it is, the RISC-OS Special – slightly delayed in publishing because we had to keep to Acorn's embargo date of the second week in January. It looks as if RISC-OS is going to be another major step forward in the life of Archimedes (and Acorn). The contributors who have already got pre-release copies seem to be of one mind about RISC-OS.

I have felt for some time that the power of the ARM chips in the Archimedes was being limited by the Arthur operating system, but it looks now as if RISC-OS will provide the means that the software developers have been waiting for to unlock that power.

The question is, will enough of the 15,000+ (?) Archimedes owners think it worth the £36 to upgrade? If they don't, the software developers won't have a big enough market to make it worth up-grading their software let alone writing entirely new software for applications that weren't possible before.

My guess is that as soon as people see just what they get for the £36 (provided they buy it straight away) they will see that it most certainly **is** worth it. In which case, I think we can see a very exciting future for Archimedes.

(By the way, £36, the Archive price for RISC-OS, **includes** U.K. postage and VAT so remember to compare like with like!)

Value for money?

Well, I've made it, it's December 20th and the magazine is on its way to the printers. Now I can have my Christmas break. The only trouble is that, because I have produced this one so soon after the December issue, there is not much in the way of Hints and Tips etc. Still, I think it is almost as full as usual because of all the RISC-OS information.

Mind you, the sharp-eyed amongst you will notice that the type-size is not quite as small as I sometimes need to use to get everything into the 52 pages. Still, I don't think anyone could complain about Archive in terms of value for money, could they?

Hope 1989 is a good year for you and thanks again for your continued support.

Paul Beverley

Archive

Volume 2 • Nº 4 • January 1989

Contents

Hardware & Software Available

• **Arc Trivia** – £24.95 from Moray Computing (or £23 from Archive) – 7,000 questions and an editor to allow you to add your own specialised questions, works on 1Mbyte machines upwards.

• **MS-DOS Shareware Disc** – Ken Biddle is compiling a Shareware disc for Archimedes MS-DOS users, so if you send him (through the Archive office) an MS-DOS formatted 3.5" disc with return packing and postage, he will send you a copy free of charge.

• **ABC Version 2** – Dabs Press have now done a version 2 of their Archimedes BASIC Compiler. The up-grade is free of charge to existing users and includes a new user guide. Should be ready by mid January. Includes double & extended precision floating point variables, RETURN is now implemented, Local error handling and various other changes.

Review Software Received...

Apart from reviews already written we have received review copies of the following software: Word Up Word Down and Star-Trader from Gem Electronics, version 1.01 of Extended precision in BASIC from Abacus.

Hints and Tips

H & T are a bit thin on the ground this month because this issue is being finished before Christmas so I get some time off and it's only a couple of weeks since I closed the December issue, so not much has come in yet. Most of the H & T that we have got were sent in by Anton Carver. Thanks Anton!

• **Using function keys** – If you want to get the function keys to produce ASCII codes, say, 200, 201, 202 etc, you use *FX 225,200 but this only works for keys <f0> (the print key) to <f9>. If you want to use <f10> upwards, you need to use *FX 221,200. This also makes <insert> into the equivalent of <f13> i.e. it generates, in this case, ASCII 213.

• **Troubles with Mitsubishi Drives?** – Yes, there are problems with some of the new 5.25" Mitsubishi drives with some of the 5.25" disc interfaces, but they can be solved. The problem is the use of pin 2 on the interface. It was not used on the BBC micro, but on the new Mitsubishi drives it is used for DC interrupt. The solution is to put a little bit of masking tape on the edge connector inside the drive mechanism so that it doesn't make contact with the p.c.b. Pin 2 is on the top of the board at the outside edge.

• **Programmable reset** – A hang-over from the BBC micro is that if you program function key 10, you will find that if you press <reset>, function key 10 will be executed! If you don't know about this, it can come as something of a shock, but it could be that you would want to make use of it if you were, say, developing a machine code program which kept hanging up the machine and forcing you to press <reset>. You could program it as *KEY 10 *BASIC|MOLD|MEDIT .|M and it would jump you back into editing the program at the point where you were.

• **Is it still going?** – If you are doing some heavy machine code programming and want to know if the computer is still working or whether you have locked it up completely, try pressing the caps lock and/or scroll lock keys and if they are still responding, your computer's not dead yet! The caps and scroll lock LED's are controlled by the computer in response to the key presses, so if they are working, it tells you that the processor is still reponding to interrupts.

• **Using ARMBE** – It is useful to be able to enter ARMBE at the point of the last error. Here is a program which sets up function key 4 so that it enters ARMBE at the point where the error occurred or, if no error then it enters where you were last editing by using “EDIT .”.

```
10 *set key$edit IF ERL=0 THEN
OSCLI (|"KEY 4 *FIXKEY4||mEDIT .
||m|") ELSE OSCLI(|"KEY 4 *FIXKE
Y4||mEDIT |" + STR$(ERL) + |"||m
|")|mMISTAKE |m*FX138,0,132|m
```

```
20 *set alias$fixkey4 set key$4
|<key$edit>
```

```
30 *fixkey4
```

• **REM’s** – Although the User Guide says that REM statements are ignored by BASIC, this is not completely true in BASIC V. This is because the block conditional IF...THEN...ELSE... ENDIF requires the THEN to be the last statement on the line. Adding a REM to the end of the line will change the flow of control, thus:

```
>LIST
 10 OK%=TRUE
 20 IF NOT OK% THEN
 30   PRINT "Wrong!"
 40 ELSE
 50   PRINT "Right"
 60 ENDIF
>RUN
Right
>LIST
 10 OK%=TRUE
 20 IF NOT OK% THEN :REM Danger!
 30   PRINT "Wrong!"
 40 ELSE
 50   PRINT "Right"
 60 ENDIF
>RUN
Wrong!
```

• **MODE3/6 gaps** – There is an undocumented difference between the way that the gaps between lines are coloured in the 25-row modes. In modes 3 and 6, the gaps are the border colour and in modes 11, 14 and 17, they are the background colour.

By experimenting with SWI"OS_ReadMode Variable", I discovered that bit 3 of variable 0 (Modeflags) indicates the condition. If the bit is set to 0, the gaps will be background colour and if 1 they are border colour, thus:

```
 10 For mode%=0 TO 17
 20   MODE mode%
 30   VDU 10,0,24,128,0|
 40   SYS"OS_ReadModeVariable"
              ,MODE,0 TO ,,modeflag%
 50   PRINT "Mode= ";MODE
 60   IF (modeflag% AND (1<<3))
         <>0 THEN t$="Border" ELSE
                    t$="Background"
 70   IF (modeflag% AND (1<<2))
                  <>0 THEN t$="No"
 80   PRINT t$;" gaps"
 90   wait%=GET
100 NEXT
```

• **Auto linefeed** – Some software packages, like First Word Plus for instance, insist that you set your printer so that it does not generate automatic linefeeds. This means that if you list a BASIC program with lines that are longer than the carriage length, the line wraps round and is overprinted. To avoid this and the difficulty of fiddling with dip switches, use the BASIC command WIDTH before printing. To do this, type WIDTH <carriage length> (e.g. WIDTH 80) before printing and WIDTH 0 to reset to the default setting after printing.

Readers’ Comments

• **Program Descriptions** – I feel Archive needs to give better documentation of how programs work, perhaps by giving structure diagrams or pseudo code? Often it is not the whole program that is of interest but some procedure that could be of interest if adapted to some other purpose. Program descriptions tend to start off describing the simple and obvious parts and then run out of space before the more complex parts are reached.

(Please regard my criticism as constructive. Archive is good value and better than .... Keep up the good work!) David Leckie, Fort William.

• **Copy Protection** – Thank you for your comments about copy protection. We had a horrendous experience because of such practices. We accidentally zapped our hard disc which contained all our accounts and although we had a back-up of the data, the accounts package was copy-protected and had already been installed on the hard disc and would therefore not install a second time. We contacted the dealer who passed us on to the software company who passed us back to the dealer. After a lot of hassle the new software arrived at the dealers who re-installed it for us. The whole process took over two weeks and cost us over £250.

Copy protection does not make sense because it inconveniences and penalizes honest people, while thieves find a way round it anyway. Why should honest users have to pay extra for a "special unprotected version" or for an additional copy? For my part, I shall give my business to those who can trust me with their software and studiously avoid software that, while supposedly attempting to prevent piracy, causes me great inconvenience and additional expense. Stanley Jebb, Dunstable.

I must say, Stanley, that I agree with you. It is my experience that, by and large, if you trust people, they respect you and do what they can to help you. Certainly, since we have started dealing in Archimedes computers and software, we have not had a single cheque that was intentionally bounced – and so we never wait for cheques to clear, we just send out the goods by return of post, in faith, so to speak.

I feel that his is linked with last month's "pig-in-the-middle" comments. Copy protection does not do anything to build up a good relationship between the software producer and the user. It would be interesting to do a survey of which software houses are most highly thought of and I suspect that, certainly from the comments I have been getting, it is those who ***don't*** *protect their software that would tend to come out on top. Ed.*

Help!!!

• **Glossary** – Would anyone be willing to write a glossary or list of common abbreviations used with Archimedes?

• **Plotter drivers** – Anyone got any software to drive HPGL plotters? Simon Lincoln, Norwich.

• **Printer drivers** – Anyone got any software to drive HP Laser Jet Printer? (I think it was someone from Meadow Computers who asked this one.)

• **Erasing arrays/variables.** Is there any way of erasing arrays and/or variables, such as the ERASE <arrayname | variablename> available in a number of other BASIC's? Roland Schaufler, W.Germany.

• **Adventure Gaming.** Alan Barclay would like to write an adventure writing system along the lines of: noun, adverb, verb multiple phrase parsing; 16k rooms, objects, messages; flags / counters; event and status tables; graphic pictures supported but no included drawing package and type format options. What Alan wants to know is whether readers would like to see such a system written and, if so, what functions they would like to see implemented. Also, would anyone like to help in testing it out when it is nearly finished. Write directly to Alan Barclay, 7 Porthill Court, Aberdeen, AB1 1DU.

• **Logistix** – Has anyone had any success importing DIFF files into Logistix? It seems that even if you export a Logistix file in DIFF format and then try to re-import it, although it says "operation complete" there is nothing in the spreadsheet when it finishes. Keith Sloan, Sevenoaks.

RISC-OS Evolution

Brian Cowan

"A computer is only as good as its operating system." So say some people – mainly those who are not prepared to write programs at machine code level. With good coding, you certainly can make a silk purse from a sow's ear; witness what is done with PC compatibles! Clearly, state-of-the-art hardware requires state-of-the-art software to facilitate its full exploitation. Hence RISC-OS for the ARM.

The old BBCs

The success of the old BBC micros is ascribed to various of its features. BBC BASIC was and is an excellent version of the language. The graphics of the machine was far in advance of its time and there were plenty of ports for interfacing/expansion; there was something for everyone. Within Acorn, the most important feature was probably the Tube. This freed the user from the limitations of the 6502 processor, allowing any microprocessor to be bolted on to the system.

Birth of ARM

Early on, there was a Z80 second processor enabling many CP/M applications to be run and the Master 512 board provided a measure of MS-DOS compatibility. To increase raw computing power, the 32016 second processor was released and the 68000 was tried within Acorn. Although good, these "bolt-ons" were not good enough; so much more could be done with the right processor. The "right processor" did not exist, so they decided to make it. Once the benefits of Reduced Instruction Set Computers were appreciated, the idea of the Acorn Risc Chip was born. Remember, Acorn has beaten IBM, Hewlett Packard and all the big boys in this area.

Dedicated Chip Set

The Acorn way was somewhat unorthodox. A set of four chips was designed: ARM, MEMC, IOC and VIDC. They have to go together in a prescribed way; versatility is sacrificed for simplicity. This is in contrast to the practice of the large chip manufacturers where new members of a microprocessor family are introduced, providing different facilities. It is likely that future Acorn products will have internal hardware almost identical to the present Archimedes range.

Fortunate Misfortune

It was not originally planned for the Acorn Risc Computer to be a BBC machine. It was to be a new machine with its own operating system, ARX, designed to fully exploit the sophisticated new hardware. Although not truly UNIX, it was designed to have many UNIX-like features. The problem was, it never got finished! In desperation, a group of Acorn employees got together to produce a BBC-like operating system in record time so that the newly developed hardware could actually be sold. The BBC gave its blessing and thus the Archimedes and Arthur were born. (For those interested, ARX has been dumped and very shortly Acorn will be releasing an "Archimedes" running real UNIX.)

Finally RISC-OS

In the rush, a number of things were overlooked/omitted. The RAM filing system was not completed, the serial port had "problems" and the desk top, while impressive as a BASIC program, was not a lot of use in practice. This was the price you had to pay for getting your hands on the machines as soon as possible. RISC-OS, now that it is here, removes most of these short-comings. There is even a form of multi-tasking implemented; its true value is discussed elsewhere in this issue. It may be the intention of Acorn to convince us that RISC-OS is an entirely new product, but in practical terms it is the latest version of Arthur.

Change of image

So why the new name? This must be seen in the context of Acorn's marketing strategy. Clearly they have decided on a change of image; the "frivolous" names such as Arthur and Podule have given way to the more sober RISC-OS and Expansion Card. Acorn now wish to be seen as a serious manufacturer of serious goods.And why not? This **IS** a serious machine; it's not just a toy. A

Bye Bye, Arthur!

Gerald Fitton

To say that Acorn's new RISC-OS adds multitasking to Arthur's version of the Archimedes OS is a gross understatement of the effect of this "upgrade". When you plug the four new RISC-OS chips into your Archimedes you will completely transform it into a brand new machine with a magnificent range of extra features. Acorn have managed to do this without prejudice to any of the software that runs under Arthur. For spending £36, you get what is, in effect, a new and much more powerful machine, together with some really good software, and this, to my mind, is a real bargain.

Of course, you will feel the real benefits when the software houses put applications on the market that make use of the new features of RISC-OS. All the indications are that Acorn are encouraging software suppliers and users alike that RISC-OS will be the standard to which both applications software and machine must conform. This makes good marketing sense since, if two standards exist then the lowest common denominator, Arthur, will be chosen by software suppliers and this will inhibit the development of applications which makes the best use of the new facilities and applications which could not be run under Arthur. So, for me and many others, it's "Bye Bye, Arthur".

First Impressions

Gone are the somewhat garish colours of Arthur's Desktop. Even before pressing any of the mouse buttons, the choice of tones and colours for the new Desktop give a much more professional feel. Having used Digital's GEM and Apple's Mac for file commands such as "copy", "rename", "delete"' etc. I found the RISC-OS Desktop similar but generally easier to use and faster. Gone are nearly all the problems associated with *configure provided the software is written to make use of the dynamic allocation of memory by RISC-OS. You can call up a display which shows you the current uses made of memory. You get told if you are trying to squeeze too many pints into the pot and are offered the chance of shutting down those tasks that are using up too much of your auto-configured memory. BASIC programs run noticeably more quickly and have more commands particularly for drawing arbitrary curved lines using Bezier curves. If you have enough memory, you can use the task display package to configure some of it as a RAM disc. This is particularly useful for fast access of data or single disc copying. Finally, there are new screen modes and a new filing system format that overcomes the "can't extend" and compact problems for ever.

Printing and Screen Dumps

One of the aims of the new RISC-OS philosophy is that many applications should use and share the same utilities such as modules and printer dumps. This philosophy has been applied to the whole process of getting from a screen display to a printed copy, in fact, you should be able to use the same printer driver for many different applications. RISC-OS comes with two printer drivers, !PSPrinter for PostScript printers and !DMPrinter for a range of dot matrix printers. If you possess a PostScript printer then all you have to do is double-click on !PSPrinter. If you have a dot matrix printer then you double-click on !DMPrinter in the same way. An icon will appear. Click on the printer type field to select your printer. Most common types will be represented. Point to the icon and press the centre mouse button, Menu. Check that the settings are right for your system, e.g. a parallel printer: These settings will then be retained in CMOS RAM.

To print text files, drag them onto the printer icon. If you wish to print from the new ArcDraw, ArcPaint or Editor then click on the Print dialogue box or drag the icon from the appropriate dialogue box onto the printer icon. There is no need to wait while printing takes place because it is multi-tasked! Many of the new applications packages will use this philosophy to your advantage and to the advantage of the application designer. It will be up to the suppliers of new printers to ensure that they supply a printer driver that latches onto the "hooks" that Acorn have included so conveniently in their new OS.

Applications supplied with RISC-OS

The application packages supplied as part of the upgrade are worth £36 on their own. I suppose that, initially at least, Acorn intended these applications as models for software developers to follow. Some are fairly trivial; one is superb.

Maestro

Acorn's music editor has been improved and now shows the music as it is played. It is menu driven from the usual set of windows that soon become familiar to the new RISC-OS user.

AClock and ArcAlarm

This is a multi-tasked clock which can be scaled from a working icon to a full screen clock. It can be digital or analogue. Because the application is multi-tasking you can have almost as many clocks on screen at once as you like. A boon for those wanting to know the time in Tokyo, London and New York simultaneously!

Calculator

The inevitable simple four function calculator.

CPU Usage

This is an interesting display showing how hard the CPU is working on the multitude of tasks that may be running simultaneously.

Lander

This is a game in which you fly a shuttle around a three dimensional landscape whilst trying to destroy objects on the ground.

Patience

Based on the familiar card game.

Puzzle

This is the 15 tile puzzle in which you move the tiles around a 4 by 4 grid in an attempt to get the numbers in some sort of order.

Madness

All the windows displayed on the screen keep wobbling. It really could drive you mad! Whoever thought of it must be – mad that is!

Magnifier

Good fun this one. Instead of the usual pointer, you are given a magnifying glass to look at any part of the screen. It makes an interesting demonstration of the high speed processing of which RISC is capable.

65Host

Although not exactly an application, it is an improved 6502 emulator that will allow you to run old BBC programs. The main difference from the emulator provided with Arthur is that this one will emulate a standard BBC without second processor and will run programs that do such "illegal" things as accessing screen memory directly instead of through the recommended VDU calls. *(Chucky-Egg, here we come!)*

ArcEdit

This is the first of the three major applications supplied with RISC-OS. It is not intended as a wordprocessor but more as a text editor that can be used to create source code, Command or Obey files.

ArcPaint

This is a multi-tasked version of Acorn's earlier sprite editor. You can have several sprites on the screen at the same time and move or copy sprites form one window or file to another. Since RISC-OS is almost mode independent, sprites created in one graphics mode can be used in any other. The lower BBC modes, such as mode 0, still shows the sprites, but the colours are replaced by shading patterns. Sprites created in ArcPaint can be dragged into ArcDraw and scaled as required.

ArcDraw

I have saved the best until last. This is a superb piece of software. It would take too long to describe all its features. Here are a few. Objects consist of either (a) text, from any fancy font, at any size, with different vertical or horizontal scales if required, (b) imported sprites which can be scaled and rotated, or (c) lines which can be straight or any arbitrary Bezier curve. This latter type of object can be edited in real-time (you see it happening as you do it) allowing you to adjust the curve to any arbitrary shape. You can add extra points if you need to do so, form reflections and rotate them through any angle. A selection of objects can be grouped so that they all are moved, scaled or rotated together. You can create multiple windows and drag objects or copies of them from one window to another. This, together with the printer drivers, must be worth the £36 on its own!

Summary

RISC-OS is not just an upgrade, nor is it just a multi-tasking version of Arthur. Plugging in the four chips gives you a new, much more exiting machine. In order to keep the Archimedes out in front of the competition, Acorn want application package designers to use all the new power of RISC-OS. In turn this means that to avoid the "lowest common denominator" effect they have to persuade a large proportion of existing users to upgrade to the new system. They have put together a marketing package that will be difficult to refuse. In addition to running the new software such as the ArcDraw package, RISC-OS will run the old BBC packages, the PC emulator and practically all Arthur designed packages. Since what you get for £36 is virtually a new machine with a most exiting future, you will have to have some very good reason indeed to make you decide to stick with Arthur and let these new developments pass by.

RAM Discs and the E-Disc Format

Brian Cowan

At last we have a RAM disc. And there's more… There is a new 800k disc format which never needs compacting so you can forget all those map full errors, and hard discs also can be formatted to operate with this file structure.

Floppy disc formats

It may seem that things are getting a bit complicated. We already have the L format for compatibility with older BBC machine and the Archimedes-only D format of 800k. Now we shall have E format, also holding 800k of data. However there is no need to panic; it is all transparent to the user. You make your choice at the disc formatting stage and following this, everything looks after itself. In fact you don't even have to make the choice as E format is the default choice. If you do *FORMAT without specifying which format type then E format is done. This is in contrast to the old ADFS where you had to specify the format required. Of course, machines running Arthur 1.2 will not be able to read or write E format discs.

File structure

In the old style formats, files had to be stored in contiguous areas on disc. This makes for rapid and continuous data access, but the disadvantage is the need to (repeatedly) compact to close up the spaces liberated by deleting files. The alternative is to store fragments of files wherever there is a space and to keep a location table of where everything is. Efficient operation of such a scheme relies on a subtle compromise. If the size of the fragments becomes too small then the location table becomes too large and file access is slowed down. On the other hand if only fragments above a certain size are allowed then there can be wasted space.

Acorn have made a study of how these things are done in MS-DOS and in UNIX. The best points have been adopted and the bad points overcome. Speed of access is maintained by keeping a copy of the location table in RAM. Presumably the old D format will soon become obsolete.

Hard Disc Format

The new scheme extends to storage on hard discs as well, although obviously presently used Winchesters will have to be reformatted. There is a hard disc formatter program written in BASIC which is included for this purpose. For the ultimate in reliability, there is provision to soak test the Winchester once it has been formatted. An interesting feature of the new ADFS is that the necessary Winchester facility of mapping out bad sectors has been extended to floppy discs as well. In certain respects, this is rather surprising since I would have thought that anyone using discs with faulty sectors was asking for trouble. However, I think this facility will facilitate the repair of damaged/corrupted discs.

RAM Disc

This is a most welcome addition. As seasoned readers will be aware, I have frequently bemoaned the lack of a RAM disc on Arthur 1.2, so I am thrilled it is finally here. What we have is a clever system but with some limitations.

It is clever in that it is just like any other filing system. In the ADFS, you refer to a file as ADFS:0.$.DIRECTORY.FILE, and similarly, with the RAM filing system, you would use RAM:$.DIRECTORY.FILE. In this respect there is nothing new to learn. However, you must create the RAM disc space before you start. This is very easy to do from the desktop by using the task manager or you can do it with *CONFIGURE.

The main limitation is that the size of the RAM disc can not be altered while it contains data. Thus, in particular, it can not be enlarged to avert an overflow. Also, you can't create more than one RAM disc. This is a pity since, apart from the convenience of doing this, it would provide a solution to the first mentioned problem. As it is, you would have to copy the RAM disc contents to either disc or some other part of RAM. The RAM disc can then be wiped and a new enlarged RAM disc created. This would then have to be reloaded. Perhaps someone will write us a utility to do this.

Filing System Modules

If you look at the ADFS relocatable module, you will be surprised how small it is. There has been considerable rationalisation of the filing system's implementation and the business part of the code is contained in the FileCore module. This is fairly well documented and it means that it is now a relatively simple task to implement new filing systems as most of the work has been done already. It should be quite easy, for instance, to create an SCSI driver for all those spare Winchesters out there. Also this could be used for reading data from compact discs.

Matters Arising

• **Sorry Westbourne Park!** – Poor old Westbourne Park got missed off Brian Smith's underground map on the Shareware Disc 2. To correct this omission, use:

```
*DIR D*.U*
LOAD"UNDER"
41872 PROCstn(1124,2272,
                        "WESTBOURNE")
41874 PROCstn(1252,2240,"PARK")
SAVE
```

BASIC V-and-a-bit

Brian Cowan

BASIC is a language which arouses strong emotions. There are those academics who believe that the teaching of BASIC is a form of corruption of the young! Such arguments are based on matters of structure and style; BASIC programs do not allow for good structuring, which encourages bad programming style. Not so with BBC BASIC, particularly BASIC V. Devotees of style and structure regard Pascal as the language of preference, but BASIC V does almost everything that Pascal does, and more. *(I suspect I'll get a goodly postbag soon! Ed.)*

Acorn are thus to be commended on the continued inclusion of a BASIC interpreter in their microcomputer ROMs. The BASIC V 1.02 of Arthur 1.2 has now been upgraded to version 1.04 in RISC-OS. There are a small number of additional features and a larger number of fixes to bugs, although it is sometimes arguable which are which!

OVERLAY

The most significant omission from version 1.02 was an overlay facility. We have function and procedure libraries but these must be loaded into memory before a program is run. With an overlay facility things could be loaded from disc, say, when required. Way back in June, Archive 1.9, I mentioned that there was a version of BASIC V, version 1.03 which included the OVERLAY keyword. I discussed its operation there. OVERLAY is included in version 1.04 and is most welcome.

Arrays and Matrices

Many people welcomed the sophisticated array and matrix handling features of BASIC V. Unfortunately things sometimes seemed to go wrong in some of my matrix programs for no apparent reason. There were bugs in the BASIC although often it was difficult to know what they were. However programs of mine which crashed in version 1.02 ran perfectly in version 1.04. Acorn have tidied up / improved the array initialisation operations. When you DIM an array, all elements are set to zero (null for string arrays). All elements can now be changed to the same value using A()=5 or A()=a. Also you can set the elements to different values using A()=1,2,3,4..... Bugs relating to local declaration and DIMing of arrays have also been fixed. It is a pity that there is no way of initialising the unit matrix, that is, the square matrix of specified dimension where all diagonal elements are one and the rest are zero – still, you can't have everything.

Other Additions

There are some other areas where the string and array handling features have been enhanced. If you have an array of string variables then SUMLEN gives the sum of the lengths of all the strings in the array and for numeric arrays, MOD gives the square root of the sum of the squares of the elements. This is a most useful mathematical function.

There have been many changes to the error descriptions. These have been made more self-explanatory. Also there are many more exported routines from CALL. These will be described in a future article. There have been some speed ups. This applies particularly to the SYS statement, especially where no results (no TO) are required.

Use with Arthur

All in all, these enhancements result in a BASIC that is better than ever. The BASIC V 1.04 relocatable module provided with RISC-OS works perfectly well under Arthur 1.2. So if you can get hold of a copy you can run it directly. There is no separate fast version to run in RAM; instead there is a utility in RISC-OS which loads any ROM relocatable module into RAM for faster operation.

I^{2}C Serial Interface

Brian Cowan

I mentioned the I^{2}C serial bus in the Hardware column back in Archive Volume 1 Nº 11. I had been told by Acorn that they could not give me much help then, but that Arthur version 2 would contain full firmware support. Well Arthur 2 has finally materialised in the guise of RISC-OS and there is indeed facility for handling the I^{2}C bus. The RISC-OS operating system contains a module called IIC and this provides a special SWI called IIC_Control (&240). I will discuss the use of the SWI after a short explanation of the I^{2}C bus.

The Bus

This bus is a two wire serial bus that has been patented by Philips. The two lines are referred to as SDA – serial data, and SCL – serial clock. With just these two lines, data is transferred around between devices connected to the bus. Each device has a unique address, whether it is a microcomputer, LCD or LED display driver, A-D or D-A converter, memory, clock or keyboard interface. Each device can act as transmitter or receiver although obviously a display driver is solely a receiver, while a keyboard interface is only a transmitter.

Masters and Slaves

Individual devices can act either as masters or as slaves. The distinction is that a master may initiate data exchange whereas a slave may not and it is the master which generates the clock signals to permit data transfer. When that happens, any device addressed is considered a slave. There may be more than one master connected to the bus although only one would be active at any one time. There is an arbitration procedure to prevent more than one master trying to take control at the same time.

Archimedes Implementation

In the Archimedes, the two lines of the I^{2}C bus are provided by the IOC. Lines C0 and C1 correspond to SDA and SCL. Within the Archimedes, the serial bus is used only for communication with the Real Time Clock / CMOS RAM chip. Use of the bus makes for considerable simplicity in the machine hardware. The RTC/CMOS RAM chip has only eight pins! In areas where speed of data transfer is not a problem the I^{2}C bus is the answer to a hardware engineer's prayer. The problem is the availability of specialised chips to connect to the bus, although glance through Mullard's Quick Reference Guide lists devices ranging from microprocessors, microcontrollers and microcomputers, through RAMs ROMs and calendar-clocks, telephony and dialing chips, display drivers, all the way to phase locked loop synthesizers!

In principle, any one or more of these devices could be interfaced to the Archimedes. Access to the I^{2}C serial bus may be made through the podule backplane. If you don't have a backplane then you can connect directly to the backplane socket on the main board. In both cases it is pins 19 and 20 of row C which provide lines SCL and SDA. Anyone proposing to use this bus in earnest might like to bring these lines out to a screened two pin connector in one of the rear blanking plates of the Archimedes. I hope to look at using some special I^{2}C chips in future articles. I will explain my connection conventions there.

Software support

Now let's look at the software side of things. The SWI IIC_Control takes three arguments. The first, R0 contains the device address. The address is a seven bit number, bits 1 to 7.

However, to this is appended bit zero which is set for a read operation and clear for a write operation. The next argument, R1 contains a pointer. This is the address where the data block may be found, and the final argument contains the length of the data block.

After the SWI has been executed, we must know whether the transaction has been successful or not. This information is contained in the overflow bit of the ARM's status register, bit V. If this bit is zero then the operation has been successful. All registers are preserved; for a read operation the data block is updated. However, if bit V is one then something has gone wrong. Register R0 then points to an error block, and for a read operation the data block may be partially updated. An error would occur, for example, if the accessed device was not present or switched off. In such a case the error would be "No acknowledge from I^{2}C device".

Device address

The address of an I^{2}C device is partially programmable, depending on the number of address pins on the chip. The Archimedes RTC / CMOS RAM chip is a PCF8583. This has one programmable address line allowing a maximum of two such chips to be used on the same bus. The rest of the address is fixed in the chip. For the PCF8583 the address byte takes the form 101000AR. Bit zero, R, is the read / not write line. Bit one, A, is the programmable address line which depends on the logic level on the chip's pin number three. The remaining bits are peculiar to the PCF8583; they would be different for other chips in the I^{2}C family.

There should be enough information here to allow those interested started in using the serial bus to get started. As yet I have not spent much time playing with it, but when I do I will report on my findings.

Languages Corner

David Wild

New Pascal Compiler

At the Micro User Show I met David Atherton of DABS Press and he told me that they were producing a version of Pascal for the Archimedes. As with the Acornsoft version it would accept all the ISO standard plus some extensions giving access to the operating system.

David claimed that, as it was designed specifically for the Archimedes rather than being ported from another machine, both compilation and running would be quicker than the Acorn version. It would also include its own editor rather than needing Twin. As many users of the Archimedes may want Twin for other purposes this might not be much of an attraction in itself but if it is optimised for Pascal programming we might find it an improvement.

He also claimed that it would be much better documented than the Acorn version, but I told him that that wasn't much of a claim! It will be very interesting to see what the program is like although it won't be ready until the middle of next year.

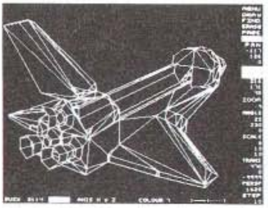
Another "silly" error message

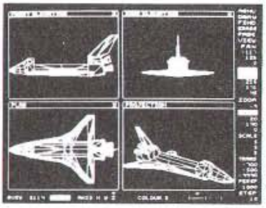
In my previous article about Acorn Pascal, I mentioned the error message you get if you make a mistake with the filename part of the 'reset' and 'rewrite' statements. The message mentions 'alias', which you haven't used. There is a similar problem if you use conformant array parameters at more than one level for the same data. When the second procedure is compiled you get an error message saying that the data types do not match – even when you have copied the definition from one procedure to another. It turns out, reasonably, that the second procedure should use a 'var' parameter for the data, and this is the error message that you get if you compile the same procedures on the BBC ISO-Pascal compiler.

SILICON VISION

SOFTWARE FOR THE ARCHIMEDES & BBC

SolidCAD





The ultimate 3D Draughting System for Architectural design, Interior design, Engineering Design and Teaching CDT. Allows drawing in plan, front & side elevations and also directly in 3D view. Includes powerful zoom & pan options for precision draughting and surface definition for creating solid colour objects. Also includes Sweep, Extrude & Macro facilities for designing very complex objects easily. Designs created with SolidCAD are compatible with the Realtime Graphics Language for high-speed flicker-free animation. The custom Archimedes version also performs smooth shading for realism. SolidCAD(Arc) users can upgrade to the Realtime Solids Modeller (Arc) for £40.00.

£49.95 (ARC & BBC) New

REALTIME SOLIDS MODELLER

The package includes both the sophisticated design environment of SolidCAD and the high speed animation capability of a Realtime Graphics Language (RGL) module developed in pure ARM Risc code for supercharged performance. The package is ideal for Architectural design, Interior design, Engineering design & teaching CDT. The RGL module can be used to create standalone flicker-free animation of designs from your own programs. Smooth shading is also performed for realistic images. Through our in-house expertise in 3D Design and High-speed techniques, no other package can rival the design environment & animation speed of the Realtime Solids Modeller.

£89.95 (ARC) New

REALTIME GRAPHICS LANGUAGE

The Realtime Graphics Language rom provides a complete 3D Solids/Wireframe animation system with 52 star commands and 3D Editors for designing objects to animate from your own programs. Includes a 35,000 pixels/sec line generator for fast 3D drawing rates. 3D Rotate, Scale, Orbit, Perspective and Turtlegraphics. Also compatible with designs created with SolidCAD (BBC).

£49.95 (BBC)

SUPER-DUMP

The ultimate printer driver which takes advantage of the highest resolution capability of ordinary Epson compatible printers to provide 1920 × 1024 resolution. Images can also be scaled, positioned and previewed before printing. Fully compatible with SolidCAD, Realtime Graphics Language, Gate Array design system & 3D CAD/Animation system. Your own graphics programs or other CAD packages can be made compatible with Super-Dump by the addition of a few simple commands. An example program is included in the package.

£15.95 (BBC), £24.95 (ARC) New

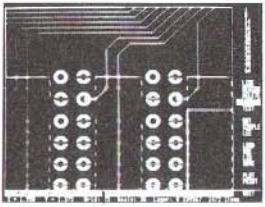
Presentation Manager

The package provides an interactive environment to create, edit and play-back computer controlled presentations for lectures and demonstrations. Also handles graph plotting for mathematical & user defined functions which can be incorporated within the presentation.

£34.95 (BBC), £49.95 (ARC) New

SILICON VISION LTD, SIGNAL HOUSE, LYON ROAD, HARROW
MIDDLESEX HA1 2AG. TEL: 01-422 2274 or 01-861 2173
FAX: 01-427 5169. TELEX: 918266 SIGNAL G.
(Access/Mastercard/Eurocard accepted)
All prices include VAT and Carriage (Overseas orders add £4).

ARC-PCB



The ultimate PCB design system developed specifically for the Archimedes with a specification that cannot be matched. Includes: Automatic routing, Rats-nesting, 8 layers, Surface mount capability, 0.001" resolution, 32" × 32" maximum board size, On-line Help, Fast Zoom/Pan/Redraw, Text & Silkscreen facility, Variable Line/Pad/Text/Grid sizes, Part Libraries, Block Move/Copy/Rotate/Mirror/Erase options, and up to 300,000 components. For hardcopy, the system supports a large number of plotters and ordinary Epson compatible printers at very high resolutions (1920 × 1024) of 240 dots/inch for near laser quality output. An entry-level ARC-PCB system (without auto-routing) is available for £99.95 (Auto-routing upgrade: £100). Enquire about our turnkey PCB design systems complete with Colour Archimedes and/or plotters with prices from £1220 to £3909 (inc VAT).

£195.00 (ARC) New

RiscBASIC

Supercharge your Archimedes Basic programs by compiling them automatically into pure ARM Risc code with the RiscBASIC compiler. Features include Relocatable modules, Cross reference of all variables, functions, and procedures, Floating point and Integer support, Stand alone code generator, Optimising compiler & Full runtime error handler.

£99.95 (ARC) New

RiscFORTH

A new 32-bit implementation of the FORTH-83 standard, designed to take full advantage of the ARM architecture. Features include Multi-tasking, Optimising compiler, built-in ARM assembler with floating point mnemonics, built-in Full screen Editor, File system interface, OS calls support, Floating point & Integer maths, WIMP support, Single-step debugger, Shadow screen for documentation, Block manipulation, Dictionary & Vocabulary display, Call finding and a standalone code generator.

£99.95 (ARC) New

See us at the B.E.T.T. Show (Stand Nº R265), Barbican, London, 18th – 21st January.

MS-DOS Column

Ken Biddle

This month we have some more information about the command codes that may be sent to the screen and the effects that they can produce. Finally, we have a review of a database package.

More about 'PROMPT'

There follows some more information on how the Dos command 'PROMPT' may be used in Dos to produce some useful effects. This information explains more fully what the 'KEY.BAT' utility was doing in a previous article and gives you some insight into how it might be extended.

You can use the 'PROMPT' command to display additional information such as which directory you are in or the date or time – in fact all of the above, if you wish. However, as you saw in the 'KEY.BAT' utility, you can also use it to redefine the function keys.

What you may not have realised is that you may also use it to redefine the function keys that produce different strings of characters when the shift, ctrl, or alt keys are depressed as well.

The format of the command is as follows:

```
PROMPT $e[0;n;"String"p
```

Where 'n' is the number identifying the key you wish to redefine and 'String' is a series of characters that you want the key to reproduce from now on.

The different strings may be produced in combination with the above keys by substituting one of the following values for 'n'

Key	Normal	[shifted]	[ctrl]	[alt]
F1	59	84	94	104
F2	60	85	95	105
F3	61	86	96	106
F4	62	87	97	107
F5	63	88	98	108
F6	64	89	99	109
F7	65	90	100	110
F8	66	91	101	111
F9	67	92	102	112
F10	68	93	103	113

As an example, you could redefine the F3 key to produce the string 'Archive!!!' by typing in:

```
PROMPT $e[0;61;"Archive!!!"p
```

A more useful example might be to display all files in the current directory with a file type of 'EXE' so that you could see what programs were on the directory. This time we will use the same function key but you should press the shift key at the same time.

```
PROMPT $e[0;86;"DIR *.EXE /W"p
```

Any function key may be returned to its original state by entering the following:

```
PROMPT $e[0;n;0;np
```

Where 'n' once again represents the number from the above table. The two key redefinitions above could be reset by the following :

```
PROMPT $e[0;61;0;61p
```

```
PROMPT $e[0;86;0;86p
```

If you are using the 'KEY.BAT' utility, you may have come across the message 'OUT OF ENVIRONMENT SPACE' this is because the 'KEY.BAT' utility first stores the original value of 'PROMPT' before it temporarily redefines it. If you have encountered this error then it just means that the environment space allocated to DOS is full. This may be avoided by putting the following line in the CONFIG.SYS file :

```
SHELL=COMMAND.COM /p /e:n
```

Where 'n' should be substituted for the number of bytes you want to allocate to the DOS environment space, e.g. to allocate 1k of environment space,

```
SHELL=COMMAND.COM /p /e:1024
```

This month's prize of a free MSDOS Public Domain disc (well, when it is finished – should be ready by the next issue) will go to the first person who sends in a new batch file or program that caters for the new function key redefinition facilities described above.

DBFast Review

There are several Database packages which run on IBM PC's and compatibles. The most widely used being DBase III from Ashton Tate. Many alternative systems based on the DBase III approach have sprung up, the most widely used being 'Clipper', 'Foxbase' and, more recently, 'DBFast'. All three differ from DBase III in that they are all compilers and purport to run faster than DBase III. This review will look at 'DBFast' since it is the cheapest and more in the price range of an Archimedes owner with a P.C. Emulator package.

DBFast is a reduced instruction set compiler based on DBase III, and is aimed primarily at application developers. As such, it greatly resembles Clipper in philosophy and practice but differs in that it lacks any bolt-on C features. There is no user interface, so it doesn't support the create and modify commands. Like Foxbase and Clipper it does offer LAN support for its applications, although this is little or no use when running on the Archimedes.

DBFast is a compiler, pure and simple. It has none of the interactive features which let you set up databases, browse through them, print lists and so on, all with help screens and menus to guide you. Instead you write DBase programs, and it converts them into machine code. Your only access to the DBase files is via the programs that you have written, or through low-level tools such as debug. In other words it's just like programming in a traditional language like 'Pascal' or 'C' but with all the advantages of the DBase language's special purpose database programming features.

Using the compiler is straight forward: you simply tell it the name of the top-level program in your DBase application (Normally the opening menu), and it goes ahead and compiles that, and all the other program modules in the 'tree' beneath it, into a single .EXE file. Before you can do that (or run a compiled program) though, you must first load the 'Database engine' program DBFast itself. The 'engine module' is a TSR (Terminate and stay resident) program which, like Sidekick, stays in the machine while other programs are loaded and run. Unlike Sidekick, however, it doesn't pop up at the press of a key. Instead it provides, via the MS-DOS interrupt system, the 'Runtime library' of subroutines which both the DBFast compiler and generated .EXE programs need in order to run properly. The fact that the 'Database engine' works on an interrupt system of operation is yet another testimony as to how good the Archimedes PC Emulator really is.

DBFast's hallmark is the small file size occupied by the runtime and compile time engines and the application files they generate.

The small file size is a particular advantage to users with only one disk drive on the Archimedes. The lucid manual explains in some detail the way DBFast uses machine memory during compiling and in the target machine.

Since memory allocation has a big effect on DBFast's performance, there is a facility to alter buffer sizes from default values. You might want to increase the allocation to speed up index operations on large single files, for instance. It automatically uses EMS memory if it is available, sadly it is not on the Archimedes.

There is a text editor supplied on the single 360k distribution disk and it is thick with commands. It offers excellent reconfigurable editing facilities, including optional word-wrap, blocking and the ability to alter the default colour combinations displayed.

The program development cycle involves you recompiling the source code every time an error manifests itself. However, this is quite painless as the compile speed is quite fast. There is also an excellent error reporting system that operates both when compiling and running programs.

The finished programs and/or applications that you write are generated as .EXE application files. You are also allowed to distribute your programs together with the DBFast database engine and DBClear which is a utility program used for removing the database engine from memory when the application running has finished. When you develop a system you would normally write a short batch file to load the database engine into memory, then invoke your application and finally execute DBClear to remove the database engine.

Performance

The package seems to be geared towards doing as much processing as possible in memory. The program is very fussy about the buffer size set up by the DOS Config.sys file and seems particularly suited to relatively small applications. When all the data the application needs can be effectively stored in memory, the database seems to run quite fast.

Some facts about the package are as follows :

- Maximum field size – 254 Characters
- Maximum fields per record – 128
- Maximum record size – 4 Mbytes
- Records per data file – 1 Billion.
- Records per Database – 10 Billion.
- Memory Variables – 64,000.
- Runtime costs – Free.
- Price – £99.00

The package can be obtained from:- Megatech, 111-113 Wandsworth High Street, London, SW18 4HY.

Conclusion

Not quite as good as Clipper, but then it is much cheaper and it does appear to run on the Archimedes OK. There appears to be a full implementation of DBase III plus programming commands and it seems to be economical on machine resources and reasonably fast if the databases are not to big on the Archimedes.

If you use DBase III at the office and would like to transfer files at home and use the data then this package offers good value for money as long as you are willing to do the programming since it has no front end, and source code must be compiled. The package is not really for beginners but you can soon get to grips with it. If you consider buying it, try and get it on trial so that you can do some extensive testing with it to see if it will meet all of your requirements. Everything appeared to work all right when I used it but then I did not destructively test every part of the package. Overall a little on the slow side with large databases but very good value for the money.

Next month

Well that's about it for this month except for an update to the software compatibility list 'IBM's STORYBOARD' appears to work with no problems. I have just received Wordstar Release 5 so by next month I should be able to tell you if that works OK.

Next month should see some more hints and tips along with a short review of some new software I have just received called 'Question Mark' which can be used to create question and answer type tutorials and automatic marking of the answers. Looks quite good.

Finally I must apologise to every one that has requested the software compatibility lists and has not yet received them. I am afraid the problem is that after going to the Micro User Show and buying lots of goodies I have been too busy playing and not doing any work. You will be pleased to know that the initial excitement is now over and I will be sending them out in the next few days.

Bye for now. See you next month… Ken.

SpellCheck for Pipedream

Gerald Fitton

Free Upgrade to PipeDream 2.2

Having sent in the registration document for my early version of PipeDream, complete with registration number (we all do this, don't we?) I received a letter from Colton Software suggesting that I take advantage of their free upgrade to version 2.2. If you haven't received one of these letters then send your original disc, having made a *BACKUP copy, together with a large stamped addressed envelope and you will get a free upgrade. If you are intending to use their new SpellCheck then you will have to obtain this new version otherwise SpellCheck won't work. If you do order SpellCheck then you do not need to send the S.A.E.

New Features of PipeDream 2.2

The new feature I noticed most was that recalculation, always fast, was now faster than ever and sorting speeds have improved. Nothing is for nothing though, and my spreadsheets with forward references seemed to need one more recalculation than they did before, still that seems to me to be a small price to pay since I always make sure with one extra recalculation before printing.

The new version of PipeDream also overcomes a real deficiency that I noticed only after my earlier review had gone to press, namely that it did not retain the original order of records during a sort when the key fields of two records were the same. The earlier version would sort on only one field at a time so there was only a very painfully long way round the problem. The new version will sort on up to ten fields simultaneously and you must do this if you wish to keep records in their original order. With this new sort you can even specify the sort on the primary and secondary fields to be a mixture of descending and ascending sorts. There is also a new standard deviation function which allows you to specify a list and a condition for selecting which members of the list you wish to include in the calculation.

The Mouse

Colton Software have provided a new module which, if you install it, will allow you to use the mouse instead of the cursor keys. I don't like it myself, and PipeDream does not provide the full Archimedes windows environment, but there are many who will use it and find it useful, particularly for marking blocks or with the pull down menus.

Installation of SpellCheck

You must first *BACKUP your original (V 2.2) disc and then *COPY to your backup all the files on the SpellCheck disc. Follow this with *instsp (not *install which I did by mistake and got just PipeDream with no SpellCheck!) and follow the instructions. Nothing could be easier! If you are familiar with PipeDream then you will notice that the extra word Spell has appeared on the top line. The new facilities can be selected starting with <Alt-S>. There are fifteen new commands on the pull down menu.

How it Works

<Alt-S-A> will auto check as you type. I am doing this now as I type this review. I have the 80,000 to 90,000 word dictionary on hard disc so I hardly see anything happen except an occasional flash of the hard disc drive light when I type in a word that I haven't used before. I have tried it with the floppy disc drive and, because it makes more noise, the presence of SpellCheck is more noticeable but it does not cause any problems with the screen display such as having to type blind whilst the display catches up. If you type in a word such as "recalculation" which does not appear in the dictionary then you get the usual "bong!" sound. If you want to check the word, then press <Alt-S-B> (for Spell Browse)

and you will be able to browse through the dictionary for the correct spelling. After browsing, you can return to the text either by pressing <escape> or, if you press <return>, a word selected from the dictionary will be inserted at the cursor. You have to delete the misspelt word with <shift-f4> or letter by letter. If you have enough spare memory, say with a 440, then you can use <Alt-S-L> to load the dictionary into memory and avoid all disc accesses.

User Dictionaries

<Alt-S-N> will allow you to create a user dictionary. SpellCheck will check this dictionary as well as the one supplied before reporting that it cannot find the word. However, you can only browse through one dictionary at a time so your own words will not appear if you use the main dictionary for browsing. Words must be less than 32 characters and begin with a letter even if you include hyphens and numbers later in the word, such as in "catch-22". The command <Alt-S-L> will let you load your user dictionary into memory; this is worthwhile if you are using a floppy disc since user directories seem to be noticeably slower. If you have more than one user dictionary then these can be merged.

Checking the Complete Document

<Alt-S-C> will check through the whole document, or just a marked block, stopping and highlighting any word not found in the dictionary. You can browse through the dictionary and alter any word that you suspect or add it to your own user dictionary. SpellCheck is quite intelligent in that, once you have accepted a word, even if you don't save it to your user dictionary, it will not ask you to check it again when it appears for a second time in the document.

Advanced Browsing

Browse has a ^? single letter or ^# many letter wildcard facility. You can browse without a document loaded so this may appeal to those of you who need help with crossword puzzles. I tried "a^?a^?^?e^?a" and SpellCheck took 20 seconds to find "anathema". A harder combination, "^?s^?e^?^?^?k", took six minutes to find "asterisk" as the only possibility. You can also find anagrams of words: "later" has five anagrams, found in less than a second but "moreter" took about two seconds to return "remoter". You can not have wild cards in anagrams. *(You may be interested in a comparison: SpellMaster took 0.02 second(!) to do each of the crossword tests and 3.6s and 4.4s for the two anagrams. Ed.)*

Start Up Options

You can save a range of user dictionaries, the column structure, page layout, printer file and colour settings in an initialisation file which will be called whenever you start up PipeDream. I found this facility, available on the original PipeDream, most useful.

What is missing?

Naturally you will not get a grammar check, so both "their" and "there" will be accepted as will "dependent" and "dependant" or "effect" and "affect". There are some strange omissions from the dictionary such as the widely used Americanism "program": the spelling "programme" is accepted. Another missing word is "wordprocessor". At least "color" is rejected in favour of "colour" although "center" and "centre" as well as "disk" and "disc" are accepted. SpellCheck does not include a Thesaurus so you cannot find a word with a meaning similar to the word you are looking up. Perhaps that is an enhancement still to come.

Summary

At about £50.00 you may not believe you have such an overwhelming need of a spelling checker that you will rush out and buy one. However, if you do have PipeDream and you or anyone in your family have a lot of word processing to do then you will find that those awkward words like "occurring" and "maintenance" will stop being the problem they so often are. SpellCheck has probably the largest general dictionary of any wordprocessor application, and the ease with which user dictionaries can be added simplifies the task of any specialist. I prefer to use PipeDream for all my non-graphics wordprocessing, especially those which also require the calculations or tabulation of a spreadsheet: because of this, the spelling checker enhancement is a "must" for me. PipeDream is also available for the PC compatible range of machines as well as the Z88 and, under the name View Professional, for the earlier BBC machines. SpellCheck is available as Archimedes and PC versions. Files are transferable between machines so you can even create a file on the Z88 on the train and then SpellCheck it when you get home on the Archimedes before printing it. What more could you want?

(I've tried using Z88 with the PC LinkII and Archimedes Pipedream and it's ever so easy. Simply type □L on the Z88 then on the Archimedes, ask to load Z:)*

Pipedream – A Personal View

Tony Cowley

Pipedream is a combined spread sheet and word processor program. You get, as it were, two for the price of one. There is now also a spelling checker which links in and bleeps at you if you type a word it does not recognise – a life saver to someone like me who has always had problems with spelling.

The manuals are well written, well indexed and clear, but the program offers so much that it is only with practice that full advantage can be taken of all the facilities provided. I therefore propose to write this review as a beginner's guide to Pipedream, and include in it the route I took to get it to work for me.

I help to run a small building company, acting as the paper pusher and linkman. This means that the first requirement was letter writing and order writing. I installed Pipedream according to the manual, which is not difficult, and promptly ran up against a practical problem. After installation, on a backup copy of course, you will find there are a number of commands needed to to activate the program. The solution is an EXEC file containing these, so all you have to type is EXEC INIT or whatever you want to call the string of commands. This presents you with a clear screen, across the top of which is a blue coloured, one line thick bar. Below this are row of letters and, on the left, numbers. These are the row and slot references, but I will come to these later on. In this bar are the following titles with red first letters. "Files Edit Layout Print Blocks Cursor Spell F1=Help". Spell is only there if you have the spell checker also installed, and F1 = Help disappears to be replaced by the file name if you are working on a named file. You also get a card to label the function keys F1 to F10, since pressing these produces special effects such as deleting rows or editing expressions.

Having got this far the user must realise that the list on the blue bar is a list of windows that can be displayed, each giving a menu of actions to operate on the text on the screen. Thus the Files menu, for example, allows saving and loading of files, the Print menu sending output to a printer, while Layout is layout. You can carry out the action you want by either pulling down the correct menu, or typing a key sequence. Pulling down a menu is easiest for the beginner. As time goes on the key sequences start to register, and the ones most useful to you can be used.

Typing a letter is just a matter of typing in the text and allowing the software to adjust the spacing. In the default setting you do not have to worry about carriage returns as this is done for you. At first, the text will not be justified, but this is switched on by the file options facility. For editing, you use the cursor keys if you are inexperienced, but you wil slowly begin to use the predefined keys and the editing window as confidence grows since these tend to be quicker. You can then graduate to move blocks of text and split lines and all sorts of fancy stuff. The facilities are there. "Blocks" is a method of marking text to allow manipulation. You should also experiment with moving the margin right and left.

In practice, there are several short cuts that Pipedream allows you to make, and these are not immediately obvious to the newcomer. First I use one or two blank files with address and date already on, so you load the one you want, and that part of the letter including the date picked up from the Archimedes' real time clock is there ready. Initially I did not realise that it was possible to use a special sequence @D@ to pick up the Arch date, but type this where you want the date and, Hey Presto, there it is ready and waiting for you. These files are saved with the required margin and header layout, and of course the justification facility included.

Macro Files

Another revelation was the macro file facility. If there are a number people you always write to, you can construct a Macro file with their address. Call this file by some recognisable name such as 'Bank' or 'Solctr' and load it at the top of the letter where a business letter normally contains the recipient's address and with window envelopes you will only need to type an address once in the future. Finally, and this will appeal to the idle, I discovered you can redefine, and save to a file called $.piped.key, definitions for the undefined keys F11 and F12, and any other keys you feel you might not need. Thus F11 is 'centre' on my keyboard and <ctrl-F12> is Centre Yours Sincerely. The file $.piped key is automatically scanned and keys defined as Pipedream boots up.

Loading and Saving

The saving of files and loading is straightforward, and Pipedream will not allow you to load a new file over an old one without asking you if you wish to save the old one first. A facility that on one or two occasions I was heartily grateful for.

Printer Drivers

There are two printer drivers supplied; Epson FX80 and JUKI 6100, but facilities are provided for editing these to suit almost any other serial or parallel printer you can think of. The process of editing is a little confusing at first but soon becomes clear with use. There is also a facility for mail shots and what are now called 'personalised' letters. I have not had need to use this but I briefly tried it out and it seemed straightforward enough. When printing, the correct printer driver has to be loaded and this has to be done every time. The margins top and bottom, and at the sides are saved with your blank file, so unless there are changes you won't need to tinker around with them after the initial settings. I did not find out until several weeks had passed that the 'PRINT' key short cuts several key strokes on printing, but once you know it's a big time-saver.

Using the Spreadsheet

Having got the letter writing end of Pipedream working nicely, I turned my attention to the spread sheet with a table of costs for a new house. So many doors at this much, so many windows at that much and so on. If you are unfamiliar with this type of program start with a small Mickey Mouse table. All spread sheets are a matrix of slots or pigeon holes into which you can put numbers, expressions or text and Pipedream is no exception. You will notice the screen is framed with A, B, C, D, E and F as the column names across the top and row numbers on the left. Any slot is known by its column letter and row number such as C21. So, for a table of costs, you will have to enter the number of windows in say column B, the price in column C and the total cost as the product of columns B and C in D.

Moving round the table is a bit confusing at first. Use <tab> and <shift-tab> to go left and right, and the up and down cursor keys to go up and down. Now put your cursor on the slot you want to fill, and press <F2>. This illuminates the slot and typing in the figure you want followed by <return> places the value in the slot. So if you have six doors at £12, put 6 in B3, 12 in C3 and then the expression B3*C3 in D3. As soon as <return> is pressed after filling D3, 36 appears in the slot. You will also notice that the position of the cursor and the slot contents are duplicated up in the top left hand corner of the screen. You can begin to see the power of programs like this. If you have got the door price wrong and it should have been £14, go back to the original entry, press <F2> and edit the figure 12 into 14. The result should be 42 immediately in D3.

I have gone into the initial stages in some detail since I found this the difficult part and once mastered you could begin to construct a useful table, gradually adding lines and columns and looking at the results to see if you had got it right. As confidence grows you can experiment with more complex formulae and Pipedream supplies some ready made expressions such as trigonometrical, financial and statistical ones for sophisticated spread sheets.

Replication

A second hurdle for me was the replication command. This command allows the user to repeat a slot, whether it contains a number or an expression, across a range of slots. So if you want to multiply a line of numbers by 10 and put the result in a line underneath the originals this is the command you use. There are a number of ways you can do this, but the following is the one I found easiest. Move the cursor to the slot you want to copy, and press <F3>, mark block. The slot will be illuminated. Now pull down the Blocks menu, and select the replicate option. A small box giving the copy from and copy to messages will appear. The origin slot should already be in the copy from message. Type in where you want to copy to, editing the value already there if appropriate and on pressing <return> all should be well. The replicate command automatically changes expressions and formulae so they are correct for their new positions. Now clear the marker and carry on.

There are a number of variations that can be played on the replication theme, but one of the more useful ones is when a slot reference in a formula has to remain fixed and not altered to the relative position. This is done by typing $ signs in front of the reference. So if you wish to always refer to a fixed slot say F17 type $F$17 in the formula.

Speeding things up

The default situation is automatic recalculation of the sheet every time a modification is made, but with a large sheet it is advisable to change this to manual recalculation as despite the speed of the Archimedes delays occur. This is done by looking at the 'Options' facility on the files menu, and changing to manual recalculation. Depending on the complexity of the sheet more than one recalculation may be required.

Simple book-keeping

Simple book-keeping can be done on a spread sheet, and although not a proper double entry system, it certainly allows the accumulation of costs in an orderly fashion with the columns correctly added up, and the figures and text on print out legible. The ability to insert rows and columns is a boon since inevitably items turn up that have been lost or forgotten about, and the spread sheet allows them to be slid into their correct slot without trouble. I also found after a time that I could incorporate printer control codes in the spread sheets to reduce print size for large tables.

The Spelling Checker

Lastly, my great friend the spelling checker. If this is set to autocheck it will search through its

80,000 word dictionary as you type and bleep if the word is unrecognised. You can then browse through the dictionary and find where you were wrong, or just correct the word and type on. Remember however that a mistake such as wit for with will go undetected since both are valid words. There is a facility for your own dictionary for specialist words, but this I have not tried since the building industry has seldom need for long words and four letters are often enough.

I can most certainly recommend Pipedream to all who, like me, have to run a small office and do not wish to rely on others for secretarial and bookeeping (There! It has just beeped at me and said it should be book-keeping) work. It produces good looking letters, budget forecasts, cash flows, bills of quantities and costings. It will however take time and effort to get the best out of it and to find your way about the manual which although clearly written is so packed full of information that it has to be read time and time again before it sinks in. Colton software maintain a telephone hot line, which I have used on a couple of occasions when all else failed. This proved very helpful. Many thanks to Colton Software.

WIMP Chess

Phil Snook

WIMP Chess is Disc Nº7 from David Pilling of Blackpool.

After initiation problems with !Boot and !readme and having to print-out the instructions, I enjoyed the challenge of the game. Using the mouse, one could have many and various lengths of game, either playing white or black against Archimedes or even playing Archimedes versus Archimedes. Use of the menu allowed undoing moves and learning from one's errors or even saving the position for later retrieval from disc. Not all selections from the menu were perfect, neither were all those from the icons at the bottom of the board; indeed 'HELP' from Archimedes was not forthcoming. Despite these imperfections, many happy hours are anticipated and one certainly cannot grumble at the price.

Each disc is £5.99, with the 5th one free. The other 8 in the series could be equally good. (See the list in Hardware & Software Available, Archive 2.3.)

Small Ads

Novice seeks patient help or user group in Manchester area. Else other novices – maybe the partially seeing can lead the partially seeing! Peter Warrington, 14 North Avenue, Burnage Garden Village, Manchester, M19 2WR.

Archimedes A310 colour, 1.2 OS, twin 3.5" drives plus over £400 of software and books. £1100. Ring 01–948–6314.

CCROM Podule with B-backup, 3 x 32k rams, spare mouse. Offers please. Matthew Treagus, 30 Hampton Lane, Blackfield, Southampton, SO4 1ZA. 0703–893596

Z88 with various ram & rom packs, printer cable, PC LinkII, mains adaptor, eraser £400. Ring 01–948–6314.

Archimedes Reference manuals (parts I & II) for sale £20. Excellent condition. Phone Onkar S Jagpal on 01–725–8643 during office hours.

Graphic Writer £13.50 o.n.o., also 'Habitat' computer desk £35. Ring 01–579–0607.

EMR Soundsynth V1.4 – £35, Creations – £13 RISCOS compatible. Phone Rob Browning on 0392–66255 or 0242–31589.

Current Directory Module

Mark Andrews

As a new Archimedes user coming from the IBM PC environment I am familiar with the idea of a modifiable prompt, as described in the User Guide, to provide an example of the use of the SETMACRO command.

"SETMACRO CLI$PROMPT <SYS$TIME>" will provide the current time as a prompt by re-evaluating the prompt string each time it is used, and the SYS$TIME variable is maintained by the operating system. Personally I find the most useful information to have at the prompt is the path or current directory, especially since I have a hard disk. It is surprising that Arthur doesn't provide a system variable to keep track of the current directory. I decided to find out what is involved in creating and maintaining such a variable.

The current directory is always changed, whether by a program or a command, via the OS_FSControl SWI. When a SWI is issued by a program, control is passed via a "vector" or pointer and this vector can be "claimed" or redirected to your own piece of code. It is then up to you whether you pass control on to the previous owner of the vector or "intercept" it entirely and replace the functions being requested with your own.

The logic behind my program is as follows:

• The program is entered via the claimed FSControlV vector.

• If the function requested is not 'set current directory', then pass control onto the previous owner of the vector.

• If it is 'set current directory' (i.e. R0=0), then ensure that we regain control after the directory has been changed, by providing our return address pushed onto the stack.

• On return we use the SWI OS_GBPB to "read current directory".

• Then issue the OS_SetVarVal SWI to update the variable.

Now if I am going to start interfering with the operating system by claiming vectors, I want to be sure that the memory into which I place my code is not going to inadvertently be overwritten by some application or utility, otherwise all functions provided by that SWI, and not just changing the current directory, will be disabled! The ideal way to see that your interfering piece of code is well looked after, is to package it as a relocatable module.

Other benefits of writing the program as a module are the ready made commands to install and remove it with RMLOAD and RMKILL, and predefined entry points for initialisation, finalisation and service call handler. The first two entry points are almost self explanatory: the initialisation code is called when the module is loaded and also after the RMA has been tidied and the finalisation code is entered for the OS_Module call with reason codes Tidy, Reinit, Delete and Clear, also if a module is loaded twice, the old one is killed.

The use of the service call handler entry point is not as immediately obvious. Service calls are issued by OS_ServiceCall and are identified by the service number in R1. They fall into two categories either soliciting information or action because of an unknown – e.g. unknown file type, command, *status or OS_Word. Or else informing modules of an event that maybe of interest to them – e.g. Error, StartUpFS, Reset, Keyboard handler, or Pre-reset.

CurrDirRM was initially written without a service call handler but the claimed vector was mysteriously being returned to its previous owner, which left the module useless and, what was more annoying, it was impossible to reinitialise, kill or tidy as it got an error whenever it tried to release a vector it no longer owned! I eventually saw the connection between, soft and machine resets and the "reset" vector. I used the service call handler to check for Reset and Pre-reset service calls, and set a flag to inform the finalisation code not to worry about releasing the vector.

The module, once assembled and saved to the modules directory, only occupies &13A bytes which even on a 512k system isn't too great a sacrifice. Now if you add the following lines to your !BOOT program:

```
*RMLOAD $.Modules.CurrDirRM

*SETMACRO Cli$Prompt <Curr$Dir>==>
```

it will install it and incorporate it into a prompt.

```
 10 REM > $CurrDirSRC
 20 REM Author : M N Andrews
 30 REM Version 1.04:30/8/88
 40 :
 50 REM Source code for a
 60 REM relocatable module.
 70 REM Title of the module is
                          CurrDirRM.
 80 REM The variable it maintains
                        is Curr$Dir.
 90 :
100 DIM start% &200
110 endofcode% =FN_AssembleRM
                            (start%)
120 OSCLI("SAVE $.Modules.CurrDirRM
           "+STR$~(start%)+" "+STR$
                       ~(endofcode%))
130 *SETTYPE $.Modules.CurrDirRM
                                &FFA
140 *STAMP   $.Modules.CurrDirRM
150 *RMLOAD  $.Modules.CurrDirRM
160 END
170 :
180 :
190 DEF FN_AssembleRM(code%)
200 :
210 LOCAL pass%
220 LOCAL modulehelp,moduletitle
230 LOCAL initcode,termcode,servcode
240 :
250 LOCAL ERROR
260 ON ERROR REPORT:PRINT " at
                     line ";ERL:STOP
270 :
280 FOR pass% = 0 TO 3 STEP 3
290 P%=code%
300 [OPT pass%
310 EQUD 0
320 EQUD (initcode-code%)
                ;Initialisation code
330 EQUD (termcode-code%)
                  ;Finalisation code
340 EQUD (servcode-code%)
               ;Service call handler
350 EQUD (moduletitle-code%)
360 EQUD (modulehelp-code%)
370 EQUD 0
380 EQUD 0
390 EQUD 0
400 EQUD 0
410 EQUD 0
420 .moduletitle
430 EQUS "CurrDirRM"
440 EQUD 0
450 ALIGN
460 .modulehelp
470 EQUS "CurrDirRM"+CHR$(&09)+
   "1.04 (30 Oct 1988)    Maintains
                         <Curr$Dir>"
480 EQUB 0
490 ALIGN
500 .initcode
510 STMFD R13!, {R7-R11,R14}
       ;These regs must be restored
                            on exit.
520 ADR R0, claimvector ;Use R0 as
     the work register to calculate
530 ADD R0, R0, #3 ;the return
          address keeping SVC_mode.
540 STMFD R13!, {R0} ;Push it onto
                      the stack and
550 B setvar ;branch to the setvar
                               code.
560 .claimvector
570 LDMVSFD R13!, {PC} ;If V is
       set then return immediately.
580 MOV R2, #0  ;No workspace
                           required.
590 STRB R2, flag ;Also if this is
          RMREINIT then reinit flag.
```

```
600 ADR R1, fscvector ;Point to
                  the our code and
610 MOV R0, #&0F ;specify the
                FSControlV vector.
620 SWI "XOS_Claim"  ;Claim the
                        vector and
630 LDMFD R13!,{R7-R11,PC} ;return
640 .termcode
650 STMFD R13!,{R7-R11,R14} ;These
     regs must be restored on exit.
660 LDRB R2, flag  ;Load flag
670 CMP R2, #&FF  ;&FF=>vector has
                        been reset
680 BEQ finished ;so we don't want
              to try releasing it!
690 MOV R0, #&0F ;Vector is being
                   released, it is
700 ADR R1, fscvector ;identified
                      by the code.
710 MOV R2, #0   ;no workspace
720 SWI "XOS_Release" ;restore
                           vector.
730 LDMVSFD R13!, {R7-R11,PC};If V
         is set return immediately.
740 CMP R10, #0 ;If termination
       fatal then destroy variable!
750 BEQ finished  ;Non fatal then
                  return, otherwise
760 ADR R0, variablename ;Point R0
            to the variable name and
770 MVN R2, #0  ;Negative length
                    (-1) to destroy.
780 MOV R4, #0  ;Variable type
790 SWI "XOS_SetVarVal" ;destroy
                   the variable and
800 .finished
810 LDMFD R13!,{R7-R11,PC} ;return
820 .servcode
830 CMP R1, #&45 ;Has someone
              pressed shift-break?
840 BEQ setflag
850 CMP R1, #&27 ;Or has someone
                   pressed reset?
860 MOVNE PC, R14 ;no then pass it
                   straight back.
870 .setflag
880 STMFD R13!, {R0-R4,R14} ;Save
                       working reg
```

```
 890 MOV R2, #&FF  ;flag=>vector
                    has been reset
 900 STRB R2, flag  ;store flag
 910 ADR R0, variablename ;Point R0
          to the variable name and
 920 MVN R2, #0 ;Negative length
                  (-1) to destroy.
 930 MOV R4, #0    ;Variable type
 940 SWI "XOS_SetVarVal" ;destroy
                  the variable and
 950 LDMFD R13!, {R0-R4,PC} ;return
 960 .fscvector
 970 CMP R0, #0 ;Is this a FSC
         request to set current dir
 980 MOVNE PC, R14 ;no then pass it
                     straight on.
 990 ADR R0, setvar ;Use R0 as the
         work register to calculate
1000 ADD R0, R0, #3 ;the return
          address keeping SVC_mode.
1010 STMFD R13!, {R0} ;Push it onto
                     the stack and
1020 MOV R0, #0  ;return R0 to its
                     proper value.
1030 MOV PC, R14 ;Call the vector's
                    previous owner.
1040 .setvar
1050 LDMVSFD R13!, {PC} ;If V is
        set then return immediately.
1060 STMFD R13!, {R0-R4} ;Save the
          regs we are going to use.
1070 MOV R0, #6   ;Tell OS_GBPB to
    read current directory into the
1080 ADR R2, workarea ; workarea.
1090 SWI "XOS_GBPB"
1100 STRVS R0, [R13] ;If it is set
       then replace the stacked R0.
1110 ADR R0, variablename ;Load R0
  with addr of variable name string.
1120 ADD R1, R2, #2 ;Point R1 to
      the directory name string and
1130 LDRB R2, [R2], #1 ;load R2
             with the length byte.
1140 MOV R4, #0  ;Variable type is
                           string.
1150 SWIVC "XOS_SetVarVal" ;Set the
     variable - If last SWI was OK.
1160 STRVS R0, [R13] ;If V is set
       then replace the stacked R0.
```

```
1170 LDMFD R13!, {R0-R4,PC} ;return
1180 .variablename
1190 EQUS "Curr$Dir"+CHR$(&10)
        ;Variable name terminated
                with a line feed.
1200 .flag
1210 EQUB 0
1220 .workarea
1230 EQUB 0   ;Zero byte
1240 EQUB 0   ;Length byte
1250 EQUS "NameString" ;Current
                      directory name
1260 EQUB 0   ;Privilege byte
1270 ]
1280 NEXT pass%
1290 =P%
```

Pacmania – Third Generation Pacman

Pacmania is the second Archimedes program to be released by Grandslam, their first being Terramex. The first thing to say is that Pacmania requires 1 Megabyte minimum to run. Pacmania is an updated version of Pac-Man but with two big changes. Firstly it's in 3-D and secondly Pac-Man can now jump over the ghosts!

Pac-Man, as I think everyone knows, lives in a maze and eats power pills while being chased by ghosts and eating an odd piece of fruit now and then. Having seen the ST and Amiga version at the PCshow in September I was looking forward to seeing it on the Archimedes. I can say that it a very good conversion. The game's layout on the Archimedes is the same as the Amiga which is what you would expect as a minimum at least, given the power of the Archimedes. The software has some very nice touches which make it a first class game; it has several catchy background tunes and cartoons that appear between each of the levels.

The 3-D effect works very well with fairly smooth scrolling on most levels though it is a little jerky on Level 4 (or, as they call it, World 4). The sprites are nicely drawn with good 3-D shading – Pac-Man even has a scaled shadow which shrinks when he jumps. One nice touch is that when you jump over a ghost its eyes look up at you as you go over the top.

The box states that there are 22 game levels with a replay action. You can start at any one of three worlds and if you complete a level but lose all your lives it allows you to restart from that same level – a nice feature – I soon tire of a game which always sends you to the very start when all your lives have gone. All in all this is a great game for the Archimedes and great fun.

Graphics 9/10: Sound 10/10: Playability 10/10: Value for Money 7/10

(Apologies to the author of this review. I can't find out who sent it in. Sorry! Ed.)

Deputy Professional – Comms Software

Tim Saxton

In the space of six months, the Archimedes has come from a machine with almost no communications software to something approaching a glut. The latest for review is 'Deputy' from Modem Marketing Ltd. Priced at only £24.95 (+VAT), it is very much cheaper than the rest, so how does it compare?

Deputy Examined

All of the basic functions are there, Mode 7 (Prestel or scrolling mode) and TTY terminal are the available emulators, and Hayes or Dumb (RTS controlled) modem selection. An off-line mailbox editor and the ability to configure the system with your MBX number, etc make its use by Prestel subscribers easy. The editor is rudimentary but works. A function key strip to identify the use of the red keys both when editing and on-line would be a helpful addition.

File transfers are limited to Prestel CET and X-modem, although CRC and file length options are selectable.

It is possible to record a logged-on session in a cache for later viewing off-line.

A boot-up file allows setting of all the selectable parameters to a chosen value, or they can be changed by toggling with an Alt + combination when running.

A directory of numbers and set-ups for your frequent calls is provided and works well, with a Hayes compatible modem the change from 75/1200 mode 7 to 2400/2400 TTY being almost instantaneous.

No script language is available, but logon sequences are possible.

The manual reflects the price in format (A5 stapled booklet) but it contains all you need to know, as well as some useful general data on communications, and how to wire up the connecting cable.

An addendum to the manual promises free upgrades to Deputy Issue II 'before the end of the year' and this will have ANSI terminal emulation, a better viewdata editor as well as other unspecified enhancements.

Using Deputy

Booting the unprotected disc gives a screen not unlike Arcterm and in fact the 'Alt' key + another control method and pop-up menus are very similar. This probably indicates that they both have a common operational heritage in Procomm on the IBM PC. It is very easy to use Deputy, there being very few frills and not a lot to learn. Alt+ H gives a help screen summarising all the features and other Alt+ combinations give access to them.

I found the software easy to understand and no serious bugs came to light, although a Prestel download did fail and cause the software to 'hang' on one occasion, and sometimes if a call failed or was aborted, a Hayes modem had to be taken off-line manually.

Conclusion

So what of Deputy? Where does it fit in the growing comms marketplace? Well it is not in the same ballpark as either Arcterm or Hearsay – but then neither is its price. As a basic, but fairly robust, communications package it works. You don't get many terminal emulations or file transfer protocols, and only two modem drivers, with no facility to add any more.

So if your modem isn't Hayes or RTS control compatible you can only use it as a dumb modem, with no autodial, etc. But that said, I think that with the promised upgrades it will find a place with the home user who just wants occasional simple communication facilities at a competitive price. A

Assembly Language Programming – 7

Alan Glover

This month we look at groups 2 and 3 of the ARM Assembly language.

Group 2 – Single byte/word memory access

This group contains two instructions. LDR is used to load a byte (8 bits) or a word (32 bits) from memory to a register. STR is used to store a register into 8 or 32 bits of memory. When 8 bit operation is used only the lowest 8 bits of the register is used. The remainder is padded with zeros by an LDR.

Three types of memory addressing are offered and it is by these that the group will be examined:

Pre-Indexed Addressing

```
STR(<condition>)(B)(T) <data>,
[<base>,(,(+ -)<offset>)](!)
```

```
LDR(<condition>)(B)(T) <data>,
[<base>,(,(+ -)<offset>)](!)
```

In this form of addressing, the offset (if present) is added to the base register. Then data is transferred between the resultant address and the register quoted as data.

The offset may be an immediate number, a register, or a shifted register and may be preceded by a + or – sign. The maximum offset is 4095 in either direction. The only group 1 type of rhs which cannot be used is a register shifted by a register.

The !, if present, causes 'write-back'. The resultant address produced by base + offset is stored back in the base register.

The B, if present, selects byte (8 bit) rather than word (32 bit) operation.

The T parameter, although included here for completeness, is no longer necessary – its intended use having been superseded by the architecture of the Archimedes.

E.g. LDR R0, [R1, #4]! (load R0 with word at address R1+4, then sets R1 to R1+4).

Post-Indexed Addressing

```
LDR(<condition>)(B)(T) <data>,
[<base>](,<offset>)
```

```
STR(<condition>)(B)(T) <data>,
[<base>](,<offset>)
```

In this addressing mode the offset is added to the base after the data transfer between the data register and the base register has taken place. Write-back always occurs.

The offset and the B and T parameters function as explained above.

E.g. LDR R0, [R1],#4 (load R0 from address R1, then R1=R1+4)

PC Relative Addressing

```
LDR(<condition>)(B)(T) <data>,
<expression>
```

```
STR(<condition>)(B)(T) <data>,
<expression>
```

This is a special addressing mode, where the expression (range + or – 4095) is added to the address of the instruction to get the address for data transfer. The expression must evaluate to a number – it cannot be a register or a shifted register.

The main use of this is to load data which is part of a program from within the program.

E.g. ADR R0,data (load R0 with address called 'data')

LDR R1,[R0] (load R1 from address R0)

is exactly the same as:

LDR R1,expression (the expression is the distance between the instruction and the data, e.g. data – P% (since P% is the address counter during an assembly)).

Group 3 – Multiple Register/Memory transfer

This group, which again consists of only two instructions is used for saving/loading one or more registers to/from a stack.

A stack is the standard type of temporary storage for machine code programs. Computer stacks operate on a principle called Last in – First Out. This can be visualised by putting balls into a tube sealed at one end. The first one which can be taken out is the last one inserted.

The ARM can support four different types of stack which all have two sets of names.

The first distinction is whether the pointer is going up or down as items are added to the stack. The second is whether the pointer is changed before or after the data transfer has taken place, which will affect whether the pointer is pointing to the most recently stored item, or to the next free space.

By convention, all stacks in the Archimedes are full, descending ones. Thus the pointer points at the item most recently stored and the pointer goes down as additional items are stored. This is the same as the stack in the 6502 series of CPUs.

The actual location of the stack, which is fixed on some CPUs, is held in a register, so several different stacks could be maintained at once.

The instructions are LDM, Load Multiple (for data from memory to registers) and STM, Store Multiple (for data to memory from registers).

The syntax is :

```
LDM(<condition>)<stack type>
<base>(!),{<reg list>}(^)
```

```
STM(<condition>)<stack type>
<base>(!),{<reg list>}(^)
```

The stack type can be FD, FA, ED, EA using the first group of attributes. F is full, E is empty, D is descending and A is ascending. When using this set, the same pair must always be used in LDM and STM instructions, so typically LDMFD and STMFD would always be used.

The alternate stack types are IA, IB, DA and DB. In these the I stands for Increment, D for Decrement, A for After and B for Before. One minor complication with using these is that complementary pairs must be used on LDM and STM instructions, e.g. for a FD stack LDMIA and STMDB would have to be used. However for abstract applications this set is easy to use since they describe what actually happens to the base.

For reference; LDMFD=LDMIA, STMFD= STMDB, LDMED=LDMDA, STMED= STMIB, LDMFA=LDMIA, STMFA= STMDB, LDMEA=LDMIA, STMEA= STMDB.

The base is the register holding the pointer, usually R13.

If the ! is present, the base will be updated after the operation. This will be the normal case, though an LDM without a ! can be useful for retrieving values saved earlier whilst leaving them on the stack for later.

The register list can contain any of the registers. The layout is single registers separated by commas, and consecutive groups linked by a dash. For example {R0,R2-R5,R14} would specify register R0,2,3,4,5 and 14.

Unless you are doing something special, always recover the same number of registers saved, to the same registers.

The order in which registers appear in the list is unimportant. The lowest numbered register is always STM'ed first and LDM'ed last.

If R15 is STM'ed, all bits (i.e. PC and status) are saved.

If R15 is being LDM'ed, the PC section only will be updated, unless ^ is present in which case all bits will be updated. This allows routines to end by restoring a previous value of R15, and decide whether to restore the original flags, or pass back flags which have been altered.

The ^ has another use in LDM and STM which is only applicable to non-User modes. More details can be found in chapter 7 of ARM Assembly Language Programming.

If the stack base is in the list of registers being STM'ed then the value saved is determined by the other registers being saved. If the stack base is the lowest number register being saved then its original value is saved, otherwise the value after write-back is saved.

E.g. `STMFD R13!,{R0-R7,R14}` (save R0-R7 and R14 using R13 as stack base. R13 is altered afterwards. R14 often contains a program address (see BL in the next section!))

`LDMFD R13,{R0-R7,R14}` (reload the registers, but because the ! has been omitted R13 is unaltered)

`LDMFD R13!,{R0-R7,R15}` (reload the registers, then amend R13. Note that R15 is being loaded with the value which was in R14, though the status bits will not be altered since the ^ is omitted)

Once more I've run out of space, but next month we'll finish off the instruction set by looking at groups 4 and 5.

Using Breakpoints with Debugger

Alan Glover

In response to a query from a reader, Alan Glover explains how to use the Debugger Module to test Assembly Language programs.

The Debugger module has facilities for inserting Breakpoints within code and continuing from them.

Breakpoints are a useful way of testing code. They allow you to stop the program at a chosen location and then to examine (and even alter) the registers before continuing.

The program below displays the letters A to Z on the screen. Once the code has been assembled, the Debugger is told to set a breakpoint at the location of the label 'LOOP'. The code is then CALLed and the Debugger will be entered showing the values in the registers on arrival at the breakpoint-ed location.

Once at the 'DEBUG*' prompt you can do one of three things:

a) Continue the execution.
 Type *CONT. (short for *CONTINUE)
 Press Y when prompted

b) Modify the registers then continue the execution.
 Type *MEMORYA <Location shown on register dump>
 The values shown are for R0 to R15. Modify any as desired.
 Type *CONT.
 Press Y when prompted

c) Remove the breakpoint and continue
 Type *BREAKCLEAR <location of breakpoint>
 Type *CONT.
 Press Y when prompted

```
>LIST
   10 REM >ATEST
   20 DIM Code 200
   30 FOR A=0 TO 3 STEP 3
   40 P%=Code
   50 [OPT A
   60 .START
   70 MOV R0,#65
   80 MOV R2,#91
   90 .LOOP
  100 SWI "OS_WriteC"
  110 ADD R0,R0,#1
  120 CMP R0,R2
  130 BMI LOOP
  140 SWI "OS_NewLine"
  150 MOV R15,R14
  160 ]
```

```
170 NEXT
180 OSCLI("BREAKSET "+STR$~LOOP)
190 CALL START
```

[Line 180 sets the breakpoint and line 190 calls the code]

```
>RUN
00009094 OPT A
00009094 .START
00009094 E3A00041 MOV R0,#65
00009098 E3A0205B MOV R2,#91
0000909C .LOOP
0000909C EF000000 SWI "OS_WriteC"
000090A0 E2800001 ADD R0,R0,#1
000090A4 E1500002 CMP R0,R2
000090A8 4AFFFFFB BMI LOOP
000090AC EF000003 SWI "OS_NewLine"
000090B0 E1A0F00E MOV R15,R14
```

```
Stopped at break point set at 0000909C

Register dump (stored at 01800584) is:
R0 = 00000041 R1 = 00000000 R2 = 0000005B R3 = 00000000
R4 = 00000000 R5 = 00000000 R6 = 00000000 R7 = 00000000
R8 = 00008700 R9 = 0009DFD8 R10 = 00000000 R11 = 00008000
R12 = 00009080 R13 = 0009DFC0 R14 = 03841CA0 R15 = 2000909C
Mode USR flags set: nzCvif

DEBUG*
```

[We have arrived at the breakpoint now. Note the values in R0 and R2.]

[*CONT. is used to continue from this point.]

```
DEBUG*CONT.
Continue from breakpoint set at 0000909C.
Execute out of line? [Y/<anything>]

AStopped at break point set at 0000909C

Register dump (stored at 01800584) is:
R0 = 00000042 R1 = 00000000 R2 = 0000005B R3 = 00000000
R4 = 00000000 R5 = 00000000 R6 = 00000000 R7 = 00000000
R8 = 00008700 R9 = 0009DFD8 R10 = 00000000 R11 = 00008000
R12 = 00009080 R13 = 0009DFC0 R14 = 03841CA0 R15 = 8000909C
Mode USR flags set: Nzcvif
DEBUG*
```

[The 'A' by 'STOPPED' was displayed by the assembly language program]

[Press Y to the 'Execute out of line' prompt]

```
DEBUG*CONT.
```

```
Continue from breakpoint set at 0000909C.
Execute out of line? [Y/<anything>]

BStopped at break point set at 0000909C

Register dump (stored at 01800584) is:
R0 = 00000043 R1 = 00000000 R2 = 0000005B R3 = 00000000
R4 = 00000000 R5 = 00000000 R6 = 00000000 R7 = 00000000
R8 = 00008700 R9 = 0009DFD8 R10 = 00000000 R11 = 00008000
R12 = 00009080 R13 = 0009DFC0 R14 = 03841CA0 R15 = 8000909C
Mode USR flags set: Nzcvif
DEBUG*CONT.
Continue from breakpoint set at 0000909C.
Execute out of line? [Y/<anything>]

CStopped at break point set at 0000909C

Register dump (stored at 01800584) is:
R0 = 00000044 R1 = 00000000 R2 = 0000005B R3 = 00000000
R4 = 00000000 R5 = 00000000 R6 = 00000000 R7 = 00000000
R8 = 00008700 R9 = 0009DFD8 R10 = 00000000 R11 = 00008000
R12 = 00009080 R13 = 0009DFC0 R14 = 03841CA0 R15 = 8000909C
Mode USR flags set: Nzcvif
Escape
DEBUG*CONT.
Continue from breakpoint set at 0000909C.
Execute out of line? [Y/<anything>]

DStopped at break point set at 0000909C

Register dump (stored at 01800584) is:
R0 = 00000045 R1 = 00000000 R2 = 0000005B R3 = 00000000
R4 = 00000000 R5 = 00000000 R6 = 00000000 R7 = 00000000
R8 = 00008700 R9 = 0009DFD8 R10 = 00000000 R11 = 00008000
R12 = 00009080 R13 = 0009DFC0 R14 = 03841CA0 R15 = 8000909C
Mode USR flags set: Nzcvif
DEBUG*
```

[Rather than step through all the remaining loops, or removing the breakpoint, the value in R0 is about to be changed using *MEMORYA to force the loop to terminate normally.]

```
DEBUG*MEMORYA 1800584
+ 01800584 : E... : 00000045 : ANDEQ R0,R0,R5,ASR #0
Enter new value : 0000005A
+ 01800584 : Z... : 0000005A : ANDEQ R0,R0,R10,ASR R0
Enter new value : [Press ESCAPE]
Escape
DEBUG*CONT.
Continue from breakpoint set at 0000909C.
Execute out of line? [Y/<anything>]
```

```
Z
>
```

[Execution finishes and the CALL returns to BASIC]

After using this technique you should always clear any outstanding breakpoints using *BREAKCLR.

The 'Execute out of line' prompt is a warning. Certain instructions cannot be executed away from their original position. In those cases the breakpoint must be cleared and the instruction executed in its proper place.

The instructions affected are those which work relative to the value of the program counter, namely :

Any ADR instructions (really ADD Rn,R15,#n or SUB Rn,R15,#n)

Any B/BL instructions (because the offset to branch is relative)

LDR and STR using PC relative addressing (eg LDR Rn,Address)

Other instructions using R15 as an operand (eg ADD R15,R15,#&200)

Ideally these should not be breakpointed in the first place.

If *MEMORYI is used to disassemble the code it will automatically replace the contents of breakpointed location with its original content. However a '*' will appear instead of a ':' after the operand. A

Pax – Budget Priced Art Package

Richard Millican

It seems that the graphics capabilities of the Archimedes have impressed various software writers. Several art packages have been released so far adding to the steadily growing range of software for the Archimedes.

Pax is the latest art package to be released and is a considerable improvement on some of the earlier programs. At the price it is being sold, most people might not expect a very concise package, but this is not the case.

Pax is based on mode 15 (256 colours, 640 x 256 resolution etc) and the basic screen layout is fairly well presented, being based around the familiar PC Paint-Brush design with a few extras such as a clock and screen position (which can be altered by clicking on it and moving the mouse, thus the whole picture is accessible without having to use the full screen option).

Features

The program has the usual basic drawing features that you expect to find in a paint program including Rubber Banding, Freehand drawing, Text (with various fonts and anti-aliasing features), Spray can, Stipple brush, Fill, Rectangles, Circles, Ellipses and Rubbers. On top of these it has Arcs, Sectors and Undo which many art programs miss out. One disappointing omission is that of triangle drawing. Most of the shape options can be filled in simply by selecting the appropriate icon also options such as Spray can and rubber can easily be varied in size.

Because uncompressed mode 15 pictures take up so much disc space and take so long to load and save, Julian Rockey and James Williams have also added a useful feature of compacting screens. This speeds up the whole process of loading and saving quite considerably. Although you cannot load the compacted pictures from BASIC and other applications you can 'Screensave' any screen or area into any mode, colour and graphic mode. This facility is called 'Export' and is very useful when a picture is needed for other products such as First Word Plus. You can also 'Import' pictures from any other mode. These facilities must be the most useful ones incorporated in Pax.

Another useful facility is that of Zooming in and pixel editing. From the menus (which are selected from the row of words at the top of the screen) there are two zoom facilities, x16 and x256. The strength of these facilities is that when they are selected it is possible to scroll around the entire screen and edit any bit.

Various Sprite routines have also been incorporated into the program. The main ones being the facility of Masking sprites (which my copy of the Sprite Editor just refuses and crashes !), Loading and Saving, Grabbing them from any part of the screen and drawing with them.

The colour bar at the bottom is a very clever piece of programming. The main bar has all the 256 colours which are available, but as well as this you can scan a picture and produce the first 16 colour that it finds, select a set of colours such as all the reds or all the greens, load a previously saved set of colours, and many other things. There is also a pattern mode available which allows you to use the pre-defined patterns or to edit a pattern and produce your own. (A small bug I noticed was in the filling with patterns that had black in them – the computer just filled one line leaving the rest of the area unchanged).

Various other facilities include execution of Star commands (but none which corrupt the memory though!), Printer drivers (on my version I had trouble with the Epson-LX one but I'm sure this will have been cleared up by now), Distort-ion (which is quite good fun to play around with), Squeezing (which squeezes any area into a ball), Enlarging, Flipping and copying areas, in fact it has almost everything that can be thought of for an art package.

Conclusion

The only major complaint that I had with the program was the fact that you could not create fonts with the program even though you could use them. Admittedly you can create things called 'sprite fonts', but these are nowhere as good as 'BBCfonts' or 'Fancy Fonts'.

But all in all I feel that this is a strong, feature-packed art package and the two writers have obviously spent a fair amount of time writing and developing Pax and at under £16 I would recommend it to all users who are currently seeking a good, comprehensive art program which beats ARM paint hands down. The package is available from Z & Z Software, Brecklands, Broad Oak, Shrewsbury, SY4 3AH priced £15.95. A

Cobra – Snake Game

Tom Allen

Cobra – Snake game with a difference

Cobra, written by Richard Millican, is an adaptation of the original BBC 'Snake' game for the Archimedes. Of course, it has a couple of additional enhancements.

The basic story line as it appears in the instructions runs as follows: "While Cyril the cobra was travelling around the world he stumbled upon a lesser known kingdom of an evil drut-dwelling wizard called Keelupweed. This wizard hated snakes and when he found out that one had entered his kingdom he cast an evil spell on Cyril so that if he wanted to go home again he must eat every pill in his kingdom. To make things worse, everything that he ate increased his size a bit. Cyril was in for a bad time. But luckily you came along and you can help him on his quest home."

What you get for your hard-earned money is thirty increasingly testing yet slightly repetitive screens, all depicting our hero in various nasty situations from which only the fingers of a skilled cobra-operator can possibly save him! The game consists of the player guiding Cyril around a screen filled with various pills and other goodies all of which affect Cyril in some way; either beneficial or harmful, but usually the latter. You must also beat the clock, which begins to be a major problem in the closing screens.

There is no music during gameplay but there is a nice little medley while the high scores are being displayed and other small melodies at other stages in the game. The ending is accompanied by a small tune and a graphics sequence, with a very nice female snake! Another feature added on the disk is a tune that will run on the Archimedes' Music editor. (The editor on the Welcome Disc).

All in all, the game is a good one and probably worth splashing out £6.00 for a 3.5" disk (works on all Arcs). You can obtain a copy from: R.Millican, 10 Stokesay, Bidston, Birkenhead, Merseyside, L43 7PU. A

Logistix – Constructing a Spreadsheet

Tim Powys-Lybbe

Like all manuals, the Logistix manual has (virtually) all the information you need, but in a rather indigestible form. What it does **not** have is a series of instructions that tell you precisely what do to when setting up a new spreadsheet, and it is sometimes the simplest of things that defeat you until you find the footnote in the manual that tells you what you should have done several hours ago!

Gilt-Edged Redemption Yields

So, as a practical example, this article takes you through the steps in making up a sheet about gilt-edged stocks, Figure 1, and the yields you can obtain by investing in gilts, compared with the classic "leave your money in the building society". In order to make sense of the sheet, financially, I need first to explain the purpose of the sheet.

It does not dwell on the feature made famous by Barlow-Clowes, that the price of gilts goes up before the dividends are paid and that you can obtain an untaxed capital gain if you sell at that point. In fact that practice was eventually declared illegal (when done by a financial house). This sheet dwells instead on the traditional feature of gilts, that you can obtain a capital gain if you can purchase stock below its redemption value and then hold it to redemption. Apart from a short period when capital gains tax was first introduced, gains on gilts have never been subject to capital gains tax, nor, unless you are a trader in them, to income tax.

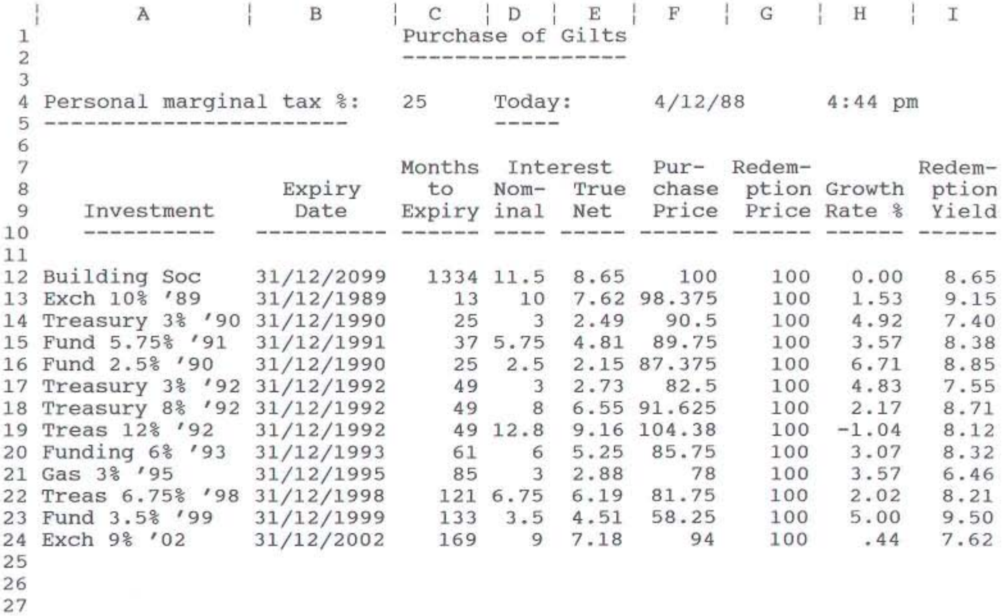
Gilts of course pay interest at a rate tied to their redemption value, or 'par' value; of course if you purchase them at a price other than par, then the interest you obtain is different on the money you have actually invested. Additionally the capital gain can be converted to a compound interest rate. In a sense you could be said to deposit a sum with a bank and have the interest reinvested each year, thus increasing the sum deposited. The redemption yield of a gilt is the sum of the interest rate, after deduction of income tax, plus the compound gain rate.

In this spreadsheet no attempt has been made to include the additional return obtained if the purchased gilt is shortly to produce a dividend, much as this may be reflected in its price. The reason for this is simple, the dividend dates are not listed in the tables of prices in the newspaper that I read.

This spreadsheet shows some of the advantages of Logistix: that it has a good range of time functions and that it has an excellent charting package built in. Without both of these, this spreadsheet would not be possible.

Constructing the sheet

This sheet consists first of the main spread sheet, shown in Figure 1, and second of the commands to convert this into a bar chart, shown in Figure 2. The first part is on the spread sheet in the range



```
  ¦       A       ¦    B     ¦  C   ¦ D  ¦  E  ¦  F   ¦  G   ¦  H   ¦  I
 1                            Purchase of Gilts
 2                            -----------------
 3
 4 Personal marginal tax %:   25     Today:      4/12/88      4:44 pm
 5 -----------------------           -----
 6
 7                            Months  Interest   Pur-  Redem-        Redem-
 8                    Expiry    to   Nom-  True  chase  ption Growth  ption
 9    Investment       Date   Expiry inal  Net   Price  Price Rate %  Yield
10    ----------   ---------- ------ ---- ----- ------ ------ ------ ------
11
12 Building Soc    31/12/2099   1334 11.5  8.65    100    100   0.00   8.65
13 Exch 10% '89    31/12/1989     13   10  7.62 98.375    100   1.53   9.15
14 Treasury 3% '90 31/12/1990     25    3  2.49   90.5    100   4.92   7.40
15 Fund 5.75% '91  31/12/1991     37 5.75  4.81  89.75    100   3.57   8.38
16 Fund 2.5% '90   31/12/1990     25  2.5  2.15 87.375    100   6.71   8.85
17 Treasury 3% '92 31/12/1992     49    3  2.73   82.5    100   4.83   7.55
18 Treasury 8% '92 31/12/1992     49    8  6.55 91.625    100   2.17   8.71
19 Treas 12% '92   31/12/1992     49 12.8  9.16 104.38    100  -1.04   8.12
20 Funding 6% '93  31/12/1993     61    6  5.25  85.75    100   3.07   8.32
21 Gas 3% '95      31/12/1995     85    3  2.88     78    100   3.57   6.46
22 Treas 6.75% '98 31/12/1998    121 6.75  6.19  81.75    100   2.02   8.21
23 Fund 3.5% '99   31/12/1999    133  3.5  4.51  58.25    100   5.00   9.50
24 Exch 9% '02     31/12/2002    169    9  7.18     94    100    .44   7.62
25
26
27
```

Figure 1 – Spreadsheet to be constructed

from A1 to I24. The second part is from A29 to O36. (I am assuming here that you know what is meant by the column and row grid references.)

Load in Logistix

Load it in according to the manual. This normally needs no more than the insertion of the disc and the usual <shift-break>. I am assuming that you have managed to get this far successfully...

Entering data or commands

What you enter from the keyboard is either data or a command. Both of these appear on the bottom line of the screen while you type them, until the data or command is complete. Data is anything that is to go into a cell on the spreadsheet. A command decides some aspect of the spreadsheet, such as column widths.

While you are entering data, you can use the 'back arrow' key (between the '£' and 'Insert' keys) to delete the last characters typed.

The Delete and Escape keys both cancel the data entry when entering data and have special functions when entering commands:

Delete: Cancels the last selection and goes back to the previous selection.

Escape: Cancels the current command altogether and reverts to the start-up condition.

The current cell

The current cell is where any data you enter will be put. It is usually lit up in a different colour or shade; additionally, you can see which it is on the line of information immediately below the sheet display itself (this is not the bottom line of the screen). There are two ways of changing the current cell. First by pressing the cursor keys (or the paging keys for coarse moves: Page Up, Page Down, F9 for left and F10 for right). Second by using the = command; to go to cell F13, say, type this command:

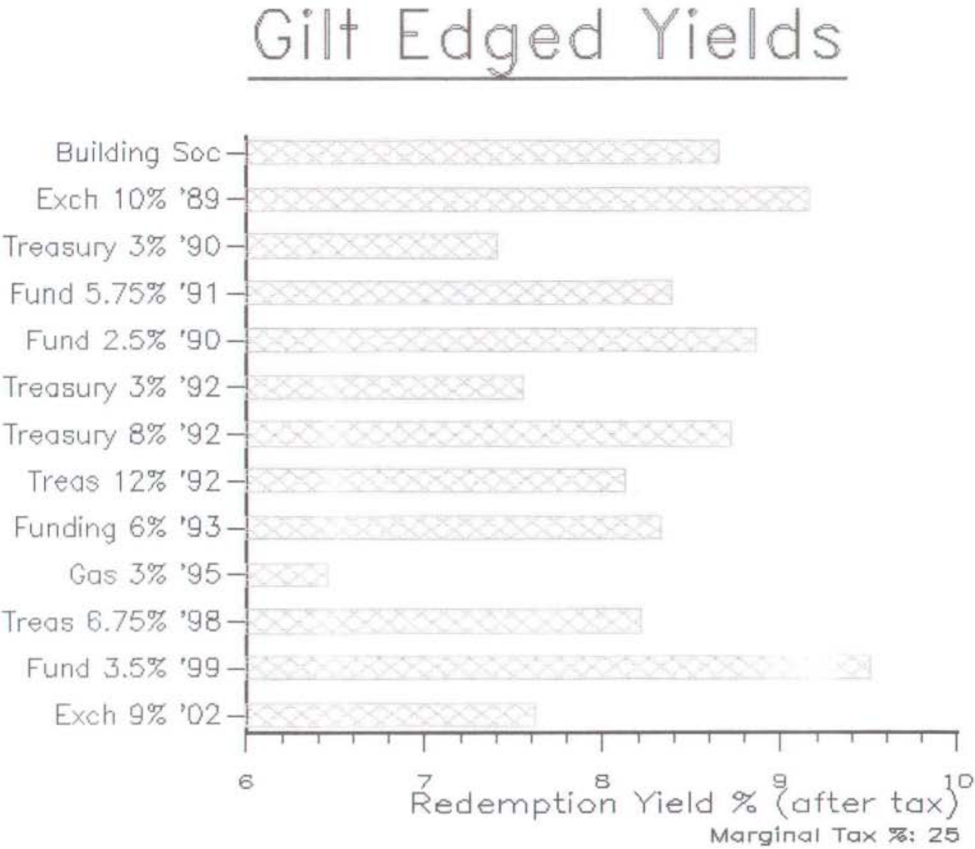


Figure 2 – Bar-chart of results

```
=F13<return>
```

This means press these five keys in succession: =, F, 1, 3 and the return key.

Slash Commands

With one exception (the = of the previous section) all commands to change the spreadsheet begin with a slash, '/'. This 'Gilt-Edged' spreadsheet needs several of these to get it into the shape of Figure 1. The following is an example of the convention I use, for the command to load in a spreadsheet with the name 'Gilt-Edge':

```
/LLGilt-Edge<return>A
```

This means that you just press the fourteen keys /, L, L, G, i, l, t, -, E, d, g, e in succession followed by the return key and finally press A – the spreadsheet file should then be loaded in. In future slash commands I will only list the keys to press; you must translate them as above.

You can, as some of you will know, put some of this information in by merely pressing the cursor and return keys, but it is almost impossible to communicate this on paper.

Preparing the sheet – Column widths

Usually this is done later, as you play about with the sheet to fit the screen. The column width is set for column A to 15 characters as follows:

1. Make somewhere in column A into the current cell. For instance, do:

```
=A1<return>
```

2. Use this slash command:

```
/FC<return>15<return>
```

The widths for the other columns are set with the same two steps. BUT use instead the following width numbers for each column:

Column	Width
B	10
C	6
D	4
E	5
F	6
G	6
H	6
I	6

Preparing the sheet – Headings

In Figure 1, the headings are shown underlined. The heading at the top is done by adjusting the cursor keys until C1 is the current cell. Then you type in "Purchase of Gilts" (without the inverted commas), press <return> and it will spill over to columns D and E. The underlines are in fact hyphens, though the underline character could be used. BUT Logistix gets confused if you just type in a row of hyphens; it thinks you have typed in minus signs. So you must put inverted commas before the hyphens, so that Logistix knows it is text:

```
"------
```

The entries in cells A4 and D7 similarly spill over into the column on the right. With D7, you should note that " Interest" has a leading space, to make it look central over columns D and E. The final step with the headings is to make them central over their columns. This has to be done in three stages, to avoid getting the wrong effect with the word " Interest". In the following slash commands, there is a problem in indicating that the down cursor key has to be pressed; I have shown it as <dn>. (The up, left and right cursor keys will be shown by <up>, <lft> and <rt> respectively.) The slash command for the first stage is:

```
/FF<up>7.C10<return><dn><dn>
<space><space><space><return>
```

Just before the last <return> this should show on the third line of the FORMAT OPTIONS on the screen "Text is centre justified".

The second and third stages are:

```
/FFD8.E10<return><dn><dn>
<space><space><space><return>
```

followed by:

```
/FFF6.I10<return><dn><dn>
<space><space><space><return>
```

Control Data in Row 4

This line has some data that is referred to elsewhere in the sheet. In C4 is the marginal tax percentage of the investor; this means the tax rate they pay on any additional income, not their average tax rate. For most people these days it is 25, though a few lucky ones may enter 40. (The instructive thing with this spreadsheet is to enter a high figure such as 75 and then see why some gilts were so attractive to top tax payers.) In this cell just enter the number, do not put a percent sign after it.

Cell F4 has one of Logistix' special functions in it. Make F4 the current cell and then type:

```
+TODAY<return>
```

The leading + sign tells Logistix that this is a numerical function.

Cell H4 has another special function, though this is not referred to elsewhere on the sheet. Get to cell H4 and type:

```
+TELTIM<return>
```

The main data rows, 12 to 24

These rows are partly data that you just enter and partly formulae. The data is in columns A, B, D, F and G. With the exception of column B, you just copy the data from your newspaper.

Column B must be in the format indicated and is entered by typing:

```
"31/12/2099<return>
```

The leading inverted commas are mandatory, otherwise Logistix thinks this is 31 divided by 12, etc. Note that for all gilts I have entered the last day of the year for the Expiry (Redemption properly) date; if you know better, please enter the correct date, it will make the sheet more accurate.

The formulae in columns C, E, H and I

Make C12 the current cell and type:

```
+(JDATE(b12)-JDATE(F4))/365*12
                          <return>
```

The b12 MUST have a lower case 'b' and the F4 must have and upper case 'F'; this is so that the formula copies correctly down the sheet, later. This is similarly important in the next three formulae.

Make E12 the current cell and type:

```
+d12*(1-C4/100)*g12/f12<return>
```

Make H12 the current cell and type:

```
+100*((g12/f12)^(12/c12)-1)
                          <return>
```

Make I12 the current cell and type:

```
+e12+h12<return>
```

At this stage you should check that it is giving the right answer in column I. If not, go back and re-enter as above.

Setting a Global Option for copying formulae

In the above formulae there are two references to row 4. The advantage of all spreadsheets is that once you have entered a formula, it need not be entered again: it can be copied to other parts of the sheet. In Logistix this is called Replication and there is a special method of allowing you to keep some cell references the same wherever the formula is copied to. It requires that the cells concerned are in Upper case in the formulae and that the first Global Option has been changed to "Adjust lower case cell references only". The Global Option is changed with this slash command:

```
/G<space><return>
```

Replicating the formulae down the sheet

The quick way of doing this is to copy all of line 12 down the page and amend the data only later. This is done with:

```
/RRA12.I12,A13.A24<return>
```

I have only used 13 rows of gilt-edged stock as this is the maximum that can be displayed with titles on one bar chart.

Protecting the formulae

This prevents accidental overwriting when amending the data for your other gilt-edged stocks. Four more slash commands:

```
/PPC12.C24<return>A
/PPE12.E24<return>A
/PPH12.H24<return>A
/PPI12.I24<return>A
```

Enter the other 12 Stocks

This is done by overwriting the data in columns A, B, D, F and G in rows 13 to 24. Make each of these cells the current cell in turn and **either** just type in the correct data **or** edit the cell with the edit slash command:

```
/E
```

The cell editor has some unusual methods of operating with special functions as follows:

Cursor Left: Go left along the cell entry being edited.

Cursor right: Go right...

Home key: Go to start of edited entry.

Copy key: Go to end of edited entry.

Cursor down: Delete character at cursor.

Cursor up: Switch from INSERT to OVERWRITE, and back again.

Return: Finish editing the cell.

Numbers of decimal places

Logistix does not have the most flexible of facilities for decimal places. The only control is either Global or by Column. You cannot set the number of decimal places by Row or by group (Field) of cells. However you can set a group of cells to Integers, zero places of decimals. This accounts for the initially strange set of slash commands below:

```
=c1<return>
/FFC12.c24<return><space>
                    <space><return>
```

This has made the 'Months to Expiry' display as integers.

```
=e1<return>
/FC<return><dn>2<return>
```

This has made column E display in two places of decimals.

```
=H1
/FC2<return><dn>2<return>
```

The latter makes columns H and I display with two places of decimals.

Check that the results are exactly as Figure 1. If they are not, then repeat the above steps until they are! Though as you have already set some of the formatting options, you may have some trouble resetting them, if necessary. When you get to the screen of options, read what is there and with the cursor, space bar and return key obtain the desired settings.

Save the sheet!

If you have not already done so (and it is prudent to save the sheet at ten minute intervals), then type this command:

```
/SLGilt-Edge<return><up>Y
```

The Y at the end of the command will not be necessary if you are saving for the first time.

That's it for now. Next month I hope to explain how to make the bar chart.

Disc Recoverer Update – 640k Discs

Richard Averill

Last month's Archive contained the unique 800k disc-recoverer. Now, here's a version that works with the 640k 'L' type. The program has been slightly modified from the one given in the December issue to cope with the old format, which is used on the BBC and Master ADFS, and can also be used on the Archimedes. Although it is more usual to use the 800k format on the Archimedes because of the increased speed and capacity, 640k is used widely in schools, and by Paul (!) because discs can be swapped between Masters and Archimedes: that's compatibility!

New Guideline

I have one guideline to add to the ones given in December's issue: keep a copy of the !Boot file in a subdirectory as well as in the root. If you have to recover the disc, then you can just copy the old boot file back into the root directory, instead of having to create it again: simple, but highly effective! It is also advisable to keep copies of files in the root directory in subdirectories, for the above reason.

RISC OS Again!

I mentioned that I would adapt the program to the new RISC OS disc format : this is not going to be easy, as the new RISC OS 'E' format looks more like an MS-DOS disc than an ADFS one, and has such things as non-contiguous files. This may increase storage efficiency, but it slows the discs down if the files are spread in bits over the disc! For the moment, anyway, it is advisable to use the 'D' or 'L' format if you want to make sure that you don't lose your files, as the RISC OS disc recoverer may not come out until April (or perhaps May) unless I receive an early copy of RISC OS!

To finish off

If you find this program useful or have some improvements to suggest, please write in! We would be pleased to hear any problems encountered. I would regard the programs as an 'insurance policy' if you ever need them. If you become more careful after having read the recovery articles, then they have been worth it. It is Sod's Law that means, in this context, that 'When recovery is possible, destruction will not occur'.

Hard Disc Recovery

I would refer you to Archive 1.12, page 12 for information on what to do if your hard disc crashes

```
 10 REM > GetDisc4
 20
 30 REM (C) Richard Averill, 1988.
 40
 50 REM Recover the directories on a
  disc with its root directory wiped.
 60 REM Runs on 310 with sizes at
          zero (except ScreenSize!),
                         and a 440.
 70
 80 MODE 12
 90 PRINT TAB(6);'"Archimedes disc
         recoverer v1.L. (C) Richard
                 Averill, 1988."'
100 PRINT "This program will re-
      create a disc that has had its
  root ($) directory over-"'"written.
       Concept by Richard Averill."'
110 PRINT "For 640k 'L' (BBC Master
                 type) ADFS discs"'
120
130 DIM surface% 640*1024
140 DIM notcopy%(150),
   directory%(150), directory$(150),
     copylist%(150),  copylist$(150)
160 DiscOp%    = &40240
170 hugo%      = &6F677548
180 OS_FSControl%=&29
190 OS_File%  = 8
200 dircounter% = 0
210 copycounter%=0
220
230 INPUT "Examine which drive ? "
                          drive%'
240 PROCprompt("Insert disc in drive
       "+STR$drive%+" and press any
                       key ...")
250
260 PROCloaddisc
270 PROCfinddirs
280 PROCfindroots
290 PROCdisplaydirs
300 PROCcreatedisc
310 END
320
330 DEF PROCloaddisc
340 PRINT '"Loading disc into memory
                                ..."
350 SYS &40240,0,1,drive%<<29,
                   surface%,640*1024
360 PRINT "Finished."
370 ENDPROC
380
390 DEF PROCcreatedisc
400 INPUT "Create a new disc in which
                    drive ? " drive%
410 IF FNask(CHR$10+CHR$13+"Drive "
     +STR$drive%+". OK ?")=FALSE THEN
                             ENDPROC
420 IF FNask("Do you wish to format
        this disc (Y/N) ?")=TRUE THEN
430   SYS 5,"Format "+STR$drive%+" L"
440   ELSE
450   IF FNask("Wipe all files on
      dest. disc (Y/N) ? ")=TRUE THEN
460     SYS "XOS_CLI","WIPE :"+
             STR$drive%+".$.* FR~C~V"
470   ENDIF
480 ENDIF
490 IF FNask("Continue (Y/N) ?")=
                 FALSE THEN ENDPROC
500
510 pre$=":"+STR$drive%+"."
520 *NODIR
530 SYS 5,"DISMOUNT "+STR$drive%
540 SYS 5,"MOUNT "+STR$drive%
550 *NOLIB
560 FOR ra%=0 TO copycounter%-1
570   spaces%=2
580   PRINT FNindent;"Dir : "+
                     copylist$(ra%)
590   SYS 8,8,pre$+"$."+
                copylist$(ra%),0,0,0
600   SYS &29,0,pre$+"$."+
                     copylist$(ra%)
610   PROCexamine(copylist%(ra%))
620 NEXT
630 ENDPROC
640
```

```
 650 DEF PROCexamine(diraddr%)
 660 LOCAL counter%,count%
 670 spaces%+=2
 680 count%=0
 690 counter%=diraddr%+5+surface%
 700 WHILE ?counter%<>0 ANDcount%<=47
 710   file$=FNmakestring(counter%,10)
 720    PROCcheck(file$)
 730    load%=counter%!10
 740    exec%=counter%!14
 750    len%=counter%!18
 760    sct%=((counter%!22)AND
                         &FFFFFF)<<8
 770    IF (counter%?3)>&7F THEN
                 type%=0 ELSE type%=1
 780    CASE type% OF
 790      WHEN 0
 800      PRINT FNindent;"Dir: "+file$
 810      SYS 8,8,file$,0,0,0
 820      SYS &29,0,file$
 830      PROCexamine(sct%)
 840      OTHERWISE
 850      type$=STR$~((load%>>8) AND
                                &FFF)
 860      PRINT FNindent;"File: "+
              file$+". Type : &"+type$
 870      SYS 8,0,file$,load%,exec%,
                 sct%+surface%,sct%+
                       surface%+len%
 880    ENDCASE
 890    counter%+=26
 900    count%+=1
 910 ENDWHILE
 920 SYS &29,0,"^"
 930 spaces%-=2
 940 ENDPROC
 950
 960 DEF PROCfinddirs
 970 LOCAL ra%
 980 FOR ra%=&C00 TO &9FF00 STEP 256
 990    addr%=ra%+surface%
1000    IF addr%!1=hugo% THEN
1010      directory%(dircounter%) = ra%
1020      ra%+=&100
1030      directory$(dircounter%) =
           FNmakestring(addr%+&4D9,10)
1040      dircounter%+=1
1050    ENDIF
1060 NEXT
1070 ENDPROC
1080
```

```
1090 DEF PROCdisplaydirs
1100 LOCAL ra%
1110 FOR ra%=0 TO dircounter%-1
1120   IF notcopy%(ra%)=FALSE THEN
1130     copylist%(copycounter%)=
                        directory%(ra%)
1140     copylist$(copycounter%)=
                        directory$(ra%)
1150     copycounter%+=1
1160   ENDIF
1170 NEXT
1180 ENDPROC
1190
1200 DEF PROCprompt(text$)
1210 PRINT text$
1220 SYS 6,21,0
1230 IF GET:ENDIF
1240 ENDPROC
1250
1260 DEF FNmakestring(addr%,maxlen%)
1270 LOCAL str$,chr%,cnt%
1280 str$=""
1290 chr%=65
1300 cnt%=0
1310 WHILE chr%>32 AND chr%<256 AND
                           cnt%<maxlen%
1320   chr%=addr%?cnt%
1330   IF chr%>32 AND chr%<256 THEN
             str$+=CHR$(chr% AND &7F)
1340   cnt%+=1
1350 ENDWHILE
1360 =str$
1370
1380 DEF PROCfindroots
1390 LOCAL ra%,ra2%
1400 FOR dir%=0 TO dircounter%-1
1410   taddr%=directory%(dir%)+
                               surface%
1420   end%=FALSE
1430   endcounter%=0
1440   FOR ra%=0 TO 79
1450     addr%=((ra%*26)+5+taddr%)
1460     IF ?addr%=0 AND end%=FALSE
       THEN end%=TRUE:endcounter%=ra%
1470   NEXT
1480   IF end%=FALSE THEN
                     endcounter%=ra%-1
1490   IF endcounter%>0 THEN
1500     FOR ra%=0 TO endcounter%-1
1510       addr%=((ra%*26)+5+taddr%)
1520       FOR ra2%=0 TO dircounter%-1
```

```
1530         IF addr%!21=directory%
                 (ra2%) THEN notcopy%
                          (ra2%)=TRUE
1540       NEXT
1550     NEXT
1560   ENDIF
1570 NEXT
1580 ENDPROC
1590
1600 DEF FNask(text$)
1610 PRINT 'text$
1620 text$=GET$
1630 CASE text$ OF
1640   WHEN "Y","y"
1650   =TRUE
1660   OTHERWISE
1670   =FALSE
1680 ENDCASE
1690
1700 DEF FNindent
1710 =STRING$(spaces%,CHR$32)
1720
1730 DEF PROCcheck(RETURN str$)
1740 LOCAL x%
1750 WHILE INSTR(str$,",")
1760   x%=INSTR(str$,",")
1770   MID$(str$,x%)="!"
1780 ENDWHILE
1790 ENDPROC
```

BASIC Program Compressor

Brian Carroll

Now that we have lots of memory for BASIC programs (don't we?), there is no excuse, as there once was, for being mean with REMs, blank lines and the length of variable names. These can make all the difference between a program that is easy to follow and to modify and one that is totally impenetrable.

Yet there is still value in compressing very long BASIC programs if you wish to store utilities in EPROM or battery-backed RAM on a ROM podule. Using the SLASH compression program which I have written, you may be able to reduce some programs by 25% or more.

Besides, a compressed program runs appreciably faster. A run of the full version of SLASH on an example 30k program including assembler took 54 seconds compared with 41 seconds using the fully compressed version of SLASH. A compressed version of the variant of SLASH (SLASHINTG on the monthly program disc) which uses the resident integers in place of normal integers runs even faster; 35 seconds for the example. Of course, it is almost impossible to read the code but this may be a useful ploy sometimes to 'protect' your ideas!

This program, SLASH, allows you to remove any or all of: redundant leading spaces and colons, REMs and empty lines. Using the BASIC Editor to do this is extremely tedious and in any case it can be hazardous because it is only too easy accidentally to concatenate lines which damage the logic flow or even to delete a line completely. Thus comprehensive testing AFTER compression is essential: a great disincentive to doing it!

Some compression programs go further than this one; e.g. they remove spaces within lines and concatenate them. This does save further space, though not all that much, but the line analysis needs to be very much more complex than I have used in this program to avoid illegal concatenation (such as in extended IF...THEN...ELSE and CASE...OF) and to preserve spaces in explicit strings. Thus such programs tend to be very slow unless they are in machine code.

In SLASH, you also have the option to enter a range of line numbers which will NOT be processed, whatever the other options set. This is useful if you want to preserve a heading, a group of explanatory lines or a fully commented procedure. You will see these lines flagged '[Skip]' during processing. The compressed program has a line added: '0REM>new-path/file-name' so the system can refer to it properly (this is not done if you select no changes!). If the old program already had a line 0 you may find another in the new program, but you can easily re-number it. The percentage compression and the time taken are reported at the end.

SLASH deals with assembler remarks (introduced by ';' or '\' as well as 'REM') but unfortunately this needs detection of entry to and exit from the assembler which is made difficult by the use of '[' and ']' within the assembler syntax and slows processing. Assembler lines are flagged whilst they are being processed. The algorithm will !!not deal correctly with remarks !!!embedded in the assembler; it assumes that they are always at the ends of lines or on separate ones, like REMs.

The program will not allow you to attempt to compress a directory or a file whose 'type' is not BASIC, or to save to an existing locked file. It will warn you of overwriting the original or any existing unlocked file, or if there is no disc in the nominated drive. In each case you can go back and correct the input. The compressed file can be put straight into the RFS by entering 'RFS:$.Library.<filename>' for the output file. <Escape> ends the program at any point without damaging files.

The program itself is 'linear' apart from the central double loop, so subroutines and procedures have not been used, just 3 functions. It uses flags extensively to control execution flow depending on the options chosen.

The new file is overlayed on top of the original in the byte array 'filespace', line by line. The original file is loaded into 'filespace', offset sufficiently for the title line (up to one screen line long) to be inserted into the new file. There are two file pointers into 'filespace' keeping note of the beginning of the unprocessed old file and of the end of the accumulating new one. There are also two line pointers; one into the section of 'filespace' holding the current old line and the other into 'linespace' in which the replacement line is being constructed. Note that BASIC lines actually BEGIN with &D but here the &D's are used as line terminators.

One other programming feature of interest is the use of 'SYS "OS_File",n' calls to allow economical but comprehensive error trapping in the file nomination routines. Note how object type, file type and access status are extracted (lines 220-250 and 320-350), and how file type is set during saving (line 1750). The SYS "OS_file" calls mostly require a POINTER in Register 1 to a path or filename anywhere in memory. In BASIC a string variable name is simply a pointer, so the 'name$' is all that need be given to the SYS call, as exemplified here.

```
 10 REM >Slash
 20 REM Brian Carroll, December 1988.
 30
 40 REM --------------------------- Initialise ----------------------------
 50 :
 60 MODE0:ON
 70 ON ERROR IF ERR=17 THEN 1790 ELSE 1800
 80 VDU19,1,6|:VDU19,0,4|:REM                       Set cyan on blue
 90 offset%=75:old_ptr%=offset%+1:in_assem%=FALSE
100 @%=&01000908:REM                                Set printing format
110 DIM linespace% &FF:REM                          Space for line processing
120 :
130 REM --------------------------- Title -----------------------------------
140 :
150 COLOUR 0:COLOUR 135:PRINTSTRING$(240," ");TAB(0,30)STRING$(80," ");
160 PRINTTAB(6,1)"SLASH - a utility to compress BASIC programs. ) Brian
                                                           Carroll, 1988."
```

```
170 COLOUR 7:COLOUR 128
180 :
190 REM -------------------------- Get input file -------------------------
200 :
210 PRINTTAB(3,5)SPC(70)TAB(3,5)"Enter input filename:  ";:INPUT""oldfile$
220 SYS "OS_File",5,oldfile$ TO found%,,lo_addr%,,file_lgth%
230 IF found%=0 VDU7:PRINTTAB(3,7)"No file "oldfile$:GOTO 210
240 IF found%=2 VDU7:PRINTTAB(3,7)oldfile$" is a directory":GOTO 210
250 IF MID$(STR$~lo_addr%,4,3)<>"FFB" VDU7:PRINTTAB(3,7)oldfile$" is not a
                                                   BASIC program":GOTO 210
260 :
270 REM -------------------------- Get output file ------------------------
280 :
290 PRINTTAB(3,7)SPC(70)TAB(3,7)"Enter output filename: ";:INPUT""newfile$
300 IF newfile$=oldfile$ VDU7:PRINTTAB(3,9)"You will over-write the input
                                                         file!"'':GOTO 290
310 ON ERROR:OFF:PRINTTAB(3,9)"Put a disc in the destination drive":GOTO320
320 SYS "OS_File",5,newfile$ TO found%,,,,,attribs%
330 ON ERROR IF ERR=17 THEN 1790 ELSE 1800:REM     Reset error handler
340 PRINTTAB(3,9)SPC(70)
350 IF found%=2 VDU7:PRINTTAB(3,9)newfile$" is a directory":GOTO 290
360 IF found% THEN
370 IF attribs% AND %00001000 VDU7:PRINTTAB(3,9)newfile$" is locked"
                                                                  :GOTO290
380 VDU7:PRINTTAB(3,9)newfile$" exists. Do you want to over-write it (Y/N)
                                                                      ? ";
390 IF FNno PRINTTAB(3,9)SPC(70):GOTO 290:REM   Start entry again
400 ENDIF
410 OFF
420 :
430 REM -------------------------- Assign space & load input file ---------
440 :
450 DIM filespace% offset%+file_lgth%+5:REM     Allow for overlapping files
460 SYS "OS_File",&FF,oldfile$,(filespace%+offset%),0:REM load old file
470 :
480 REM -------------------------- Set action flags -----------------------
490 :
500 PRINTTAB(12,11)"1. Remove REMs (Y/N)?..................... ";
510 IF FNno rems%=FALSE ELSE rems%=TRUE
520 PRINTTAB(12,12)"2. Remove leading spaces & colons (Y/N)? .. ";
530 IF FNno leaders%=FALSE ELSE leaders%=TRUE
540 PRINTTAB(12,13)"3. Remove empty lines (Y/N)? .............. ";
550 IF FNno empties%=FALSE ELSE empties%=TRUE
560 IF rems% OR leaders% OR empties% no_change%=FALSE ELSE no_change%=TRUE
570 :
580 REM ------------------------- Get range of lines not to be cut --------
590 :
600 PRINTTAB(12,14)"4. Process all lines (Y/N)? ............... ";
610 IF FNno THEN
620 PRINTTAB(12,14)SPC(67);
630 PRINTTAB(12,14)"4. Stop processing @ line ";:INPUT""begin%
640 IF begin%<0 VDU7:GOTO 620
650 PRINTTAB(43,14)", resume @ line ";:INPUT""finish%
```

```
 660 IF finish%>65279 OR begin%>=finish% VDU7:GOTO 620
 670 ELSE
 680 begin%=0:finish%=0:REM                       This looks odd, but it's OK
 690 ENDIF
 700 :
 710 REM -------------------------- Insert first line: '0REM>newfile' ------
 720 :
 730 filespace%?0=&D:REM                          Insert 1st byte, always &D
 740 IF no_change% THEN
 750 new_ptr%=1:REM                              Set pointer to the start + 1
 760 ELSE
 770 filespace%!1=0:REM                           Insert line No; hi=0, lo=0
 780 j%=LEN(newfile$):filespace%?3=j%+6:REM       Insert line length
 790 filespace%?4=&F4:filespace%?5=&3E:REM        Insert 'REM>'
 800 FOR mover%=1 TO j%:REM                       Insert title 'newfile$'
 810 filespace%?(mover%+5)=ASC(MID$(newfile$,mover%,1))
 820 NEXT
 830 filespace%?(j%+6)=&D:REM                     Insert &D for next line
 840 new_ptr%=j%+7:REM                            Set pointer to next line
 850 ENDIF
 860 :
 870 REM ------------------------ Process old file, line by line ----------
 880 :
 890 TIME=0:OFF:VDU19,1,3|:REM                    Set yellow on blue
 900 IF no_change% PRINTTAB(3,16)"Not much point, is there? ........... Oh
   well, if you insist!":REPEAT UNTIL TIME>300:REM  Message, & delay to read
                                                                          it
 910 TIME=0:PRINTTAB(3,16)SPC(76)TAB(3,16)"Processing line";
 920 :
 930 WHILE old_ptr%-offset%<file_lgth%-1:REM     Start. Ignore final &FF
 940 :
 950 linespace%?0=filespace%?(old_ptr%):REM     Transfer line number hi byte
 960 linespace%?1=filespace%?(old_ptr%+1):REM    and lo byte
 970 lineNo%=256*(linespace%?0)+linespace%?1:REM Calculate line No. and
 980 PRINTTAB(19,16)STR$(lineNo%);:REM           print it
 990 old_line_lgth%=filespace%?(old_ptr%+2):REM  Get line length
1000 :
1010 IF lineNo%>=begin% AND lineNo%<=finish% process%=FALSE ELSE process%=TRUE
1020 IF leaders% no_pads%=TRUE ELSE no_pads%=FALSE:REM Set flags on each line
1030 :
1040 oldline_ctr%=3:newline_ctr%=3:REM           Bytes 0-2 already entered
1050 :
1060 REPEAT
1070 j%=filespace%?(old_ptr%+oldline_ctr%):REM   Get next byte from old file
1080 :
1090 IF process% PRINTTAB(40,16)"      " ELSE PRINTTAB(40,16)"[Skip]":GOTO
1190
1100 :
1110 IF no_pads% THEN
1120 WHILE (j%=&20 OR j%=&3A):REM                Look for leading spaces or
1130 oldline_ctr%+=1:REM                         colons & discard them
1140 j%=filespace%?(old_ptr%+oldline_ctr%)
```

```
1150 ENDWHILE
1160 no_pads%=FALSE:REM                          Flag off for rest of line
1170 ENDIF
1180 :
1190 IF j%=&5B THEN
1200 IF NOT(in_assem%)AND FNstart in_assem%=TRUE:PRINTTAB(50,16)"[Assembler]
1210 ENDIF:REM                                   Seek start if outside assm.
1220 :
1230 IF j%=&5D THEN
1240 IF in_assem% AND FNend in_assem%=FALSE:PRINTTAB(50,16)SPC(12)
1250 ENDIF:REM                                   Seek end if inside assm.
1260 :
1270 IF NOT(process%) THEN 1340
1280 :
1290 IF rems% THEN
1300 IF j%=&F4 j%=&D:GOTO 1320:REM               Replace 'REM' by EOL (&D),
1310 IF in_assem% AND (j%=&3B OR j%=&5C) j%=&D:REM likewise ';' or '\',
1320 ENDIF:REM                                   This cuts line at 'REM' etc
1330 :
1340 linespace%?newline_ctr%=j%:REM              Add byte to end of line
1350 newline_ctr%+=1:REM                         Increment new and old line-
1360 oldline_ctr%+=1:REM                         length counters
1370 UNTIL j%=&D:REM                             EOL recognised so
1380 linespace%?2=newline_ctr%:REM               insert new line length
1390 :
1400 IF NOT(empties% AND newline_ctr%=4) THEN
1410 FOR mover%=0 TO newline_ctr%:REM            Copy new line back into
1420 filespace%?(new_ptr%+mover%)=linespace%?mover%:REM filespace on top of
1430 NEXT:REM                                    old file, then move pointer
1440 new_ptr%+=newline_ctr%:REM                  if line stored; anyway
1450 ENDIF
1460 old_ptr%+=old_line_lgth%:REM                move old file pointer along
1470 :
1480 ENDWHILE:REM                                Start on next line
1490 :
1500 filespace%?(new_ptr%)=&FF:REM               Put end flag into new file
1510 :
1520 REM ---------------------- Report results --------------------------
1530 :
1540 VDU19,1,6|:PRINTTAB(3,16)SPC(75)TAB(3,16)"REMOVED:  ";:REM back to cyan
1550 IF no_change% PRINT"Nothing.":GOTO 1610
1560 IF rems% PRINT" Remarks. ";
1570 IF leaders% PRINT"Leading spaces & colons. ";
1580 IF empties% PRINT"Empty lines.";
1590 factor=ABS(100*(file_lgth%-new_ptr%+1)/file_lgth%)
1600 IF file_lgth%>(new_ptr%+1) decr%=TRUE ELSE decr%=FALSE
1610 PRINTTAB(3,18)"Input file, "oldfile$", was "STR$(file_lgth%)" bytes."
1620 PRINTTAB(3,20);
1630 IF no_change% PRINT"Output not saved; ";:GOTO 1700 ELSE PRINT"Output
                                                        file, "newfile$", ";
1640 IF decr% PRINT"has been reduced to "; ELSE PRINT"has been increased to "
1650 PRINTSTR$(new_ptr%+1)" bytes."
```

```
1660 @%=&0102010A
1670 PRINTTAB(3,23)"This is a";
1680 IF decr% PRINT" reduction of "; ELSE PRINT"n increase of ";
1690 PRINTSTR$(factor)"%; ";
1700 PRINT"processing took about "STR$((TIME DIV 10)/10)" seconds."
1710 IF no_change% THEN 1790
1720 :
1730 REM ------------------------ Save new file, setting BASIC file type ---
1740 :
1750 SYS "OS_File",10,newfile$,&FFB,,filespace%,(filespace%+new_ptr%+1)
1760 :
1770 REM ------------------------ End properly if error or normal end ------
1780 :
1790 PRINTTAB(3,27)"Finished.":GOTO 1810
1800 PRINTTAB(3,27);:@%=&906:VDU7:REPORT:PRINT" at line"ERL
1810 PRINT:@%=&90A:ON:CLOSE#0:ON ERROR OFF:CLEAR:*FX 202,32
1820 END
1830 :
1840 REM ------------------------ Test for Y, y, N or n. Print result ------
1850 :
1860 DEF FNno:LOCAL key%:ON:*FX15,1
1870 REPEAT key%=(GET AND &DF):REM               Converts to upper case
1880 UNTIL key%=78 OR key%=89:REM                Characters 'N' or 'Y'
1890 IF key%=78 PRINT"No":=TRUE ELSE PRINT"Yes":=FALSE
1900 :
1910 REM ------------------------ Test for start of assembler -------------
1920 :
1930 DEF FNstart
1940 LOCAL k%,l%,opt$:opt$=""
1950 FOR k%=1 TO 20:REM                          Read forward 20 bytes
1960 l%=filespace%?(old_ptr%+oldline_ctr%+k%):REM Get character
1970 opt$+=CHR$(l%):REM                          Form string
1980 NEXT:REM                                    Note OPT is NOT tokenised,
1990 =INSTR(opt$,"OPT"):REM                      look for it: TRUE/FALSE
2000 :
2010 REM ------------------------ Test for end of assembler ---------------
2020 :
2030 DEF FNend
2040 LOCAL k%,l%,opt$:k%=0:opt$=""
2050 REPEAT:REM                                  Read backwards
2060 l%=filespace%?(old_ptr%+oldline_ctr%):REM   Get character
2070 IF l%<>&20 opt$=CHR$(l%)+opt$:REM           Skip spaces & form string
2080 oldline_ctr%-=1:k%-=1:REM                   Decrement counters
2090 UNTIL l%=&3A OR oldline_ctr%=2:REM          Until ':' or start of line
2100 oldline_ctr%-=k%:REM                        Restore line pointer
2110 IF opt$=":]" OR opt$="]" =TRUE:REM          Assembler end identified
2120 =FALSE:REM                                  or else a register bracket
2130 REM ------------------------ End of the program ----------------------
```

Competition Time!!

Steve Picton, I.F.E.L.

Steve has made up a competition for us and has offered a 2-slot backplane, including fan, as a prize. Many thanks, Steve. (Free advert below as a thank-you!) We'll put in a set of monthly program discs as another prize, just in case the winner owns a 440! He or she can then choose between the two prizes and the second place winner will get the other prize.

The Competition

The largest known prime number is:

(2^216091)-1

and so the task is to write a program to provide the following information;

1) How many digits long is the number? (Fairly obvious, using logs)

2) What are the first 10 digits of this number?

3) What are the last 10 digits of the number?

4) What is the total of all the digits?

5) What was the time taken for your program to produce the above answers?

(The number must be given in decimal.)

There are no limitations as to what programming language you use, but to be fair, if you use BASIC, it must be BASIC 1.02, though you can use the ram version if you like!

To cut down on postage and administration, you only need send Steve your answers on paper, though obviously the two winners would need to submit their version on disc eventually for verification.

Please send your entries to Steve at the address below by **10th February, 1989.** No entries received after that date will be counted.

Fact-File

ABACUS Training	29 Okus Grove, Upper Stratton, Swindon, Wilts, SN2 6QA.
ACE Computing	27 Victoria Road, Cambridge, CB4 3BW. (0223–322559)
Atomwide Ltd	23 The Greenway, Orpington, Kent, BR5 2AY. (0689–38852)
Colton Software	149-151 St Neots Road, Hardwick, Cambridge, CB3 7QJ. (0954–211472)
Computer Concepts	Gaddesden Place, Hemel Hempstead, Herts, HP2 6EX. (0442–63933)
Computerware	55 Romulus Court, Brentford Dock, Justin Close, Brentford, Middlesex.
Dabs Press	5 Victoria Lane, Whitefield, Manchester, M25 6AL. (061–766–8423)
David Pilling	P.O.Box 22, Thornton Cleveleys, Blackpool. FY5 1LR
EMR Ltd	14 Mount Close, Wickford, Essex, SS11 8HG. (0702–335747)
Fairhurst Instruments	Dean Court, Woodford Road, Wilmslow, SK9 2LT. (0625–525–694)
GEM Electronics	17 Tandragee Road, Portadown, Craigavon, BT62 3BQ.
Ian Copestake Software	10 Frost Drive, Wirral, L61 4XL. (051–648–6287)
IFEL	36 Upland Drive, Plymouth, Devon, PL6 6BD. (0752–847286)
Intelligent Interfaces	43B Wood Street, Stratford-on-Avon, CV37 6JQ. (0789–415875)
Impact	Unit 3, 1 Percy Street, Sheffield, S3 8AU.
Lingenuity	P.O.Box 10, Halesworth, Suffolk, IP19 0DX. (0986–85–476)
Minerva Systems	69 Sidwell Street, Exeter, EX4 6PH. (0392–437756)
Mitre Software	26 Creechurch Lane, London, EC3A 5BA. (01–283–4646)
Modem Marketing Ltd	611 Lincoln Road, Peterborough, PE1 3HA. (0733–52440)
Moray Computing	Enzie, Slackhead, Buckie, Moray, AB5 2BR. (05427–377)
Psientific Software	37 Cottesmore Road, Hessle, North Humberside, HU13 9JQ. (0482–649187)
Richard Millican	10 Stokesay, Bidston, Birkenhead, L43 7PU
Silicon Vision Ltd	Signal House, Lyon Road, Harrow, Middlesex, HA1 2AG. (01–422–2274)
Superior Software	Regent House, Skinner Lane, Leeds, LS7 1AX. (0532–459453)
Video Electronics Ltd	Wigan Road, Atherton, Manchester, M29 0RH. (0942–882332)
Z & Z Software	Brecklands, Broad Oak, Shrewsbury, SY4 3AH.

Norwich Computer Services 18 Mile End Road, Norwich, NR4 7QY. (0603-507057)

Archive

Subscription Magazine and Support Group for *Archimedes* Users

Archive Magazine contains:

- News
- Reviews
- Hints and Tips – a major feature
- Articles for Beginners
- The Latest Technical Information
- Program Listings
- Free Small Ad's Section
- HELP – Requested and Offered
- Contact Box – to help you form common interest groups

Eureka! – Bulletin Board

0603-250689 on 1200/75 or 300/300

- Very Latest News
- Down-load Software
- Mailbox Facilities
- Chat line

Technical Help Service (£8 / year)
A telephone hot-line service for immediate help with your technical problems. Any member can send written enquiries, but for a fast response use the THS!

Members' Discount: 7.5% off software from Computer Concepts, Minerva, Clares, Acorn and others when purchased through Norwich Computer Services.

Subscription: 12 issues £14.50 (UK)
Europe £20, Middle East £25,
America / Africa £28, Elsewhere £30.
Technical Help Service £8

N.B. All earlier issues have now been re-printed – you may back-date your subscription as far as issue 1 (October 1987) – to take advantage of this huge bank of information.

Archimedes is a trademark of Acorn Computers Ltd.

* Please send copies of *Archive* magazine for one year starting from
Volume ___ Issue _______

* Please enrol me on the Technical Help Service for one year. (£8)

I enclose a cheque for £ ________ payable to "Norwich Computer Services".

Name: ______________________________

Address: ______________________________

____

____________________ Postcode: __________

Norwich Computer Services, 18 Mile End Road, Norwich, NR4 7QY

Special two-year subscription starting from volume 1, issue 1 – £27